PEACH: PERSONALIZED NEGATIVE CACHING AND SAMPLING FOR UNBIASED OPTIMIZATION OF EXTREME MULTICLASS CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Extreme multiclass classification with the cross-entropy loss is computationally challenging because evaluating its normalization term requires summing over a huge number of classes. Approximating this term with a small subset of classes, as in sampled softmax, yields biased stochastic gradients that in general do not help stochastic optimization converge to a stationary point of the full objective. Hard-negative mining can reduce this approximation error, but does not remove the bias and incurs additional computational cost. Recent work removes the bias by reformulating cross-entropy minimization as an equivalent min-min problem, in which an auxiliary variable learns the normalization term and unbiased stochastic gradients can be computed efficiently. We find, however, that uniform sampling within this framework produces high-variance gradient estimates and slow convergence in practice. To address this limitation, we propose **PEACH**, an unbiased optimization method based on **efficient, personalized negative caching and sampling**. PEACH maintains a small cache of hard negative classes for each example, evaluates their contributions exactly, and uses simple random sampling to estimate the sum over the remaining classes. Our theoretical analysis shows that this strategy preserves unbiasedness while substantially reducing the variance of stochastic gradients for min-min optimization. Across datasets with hundreds of thousands of classes, PEACH substantially accelerates optimization in wall-clock time compared with existing methods. At matched training loss, PEACH achieves up to $\mathbf{2}$–$\mathbf{10}\times$ speedup compared with biased and unbiased optimization methods, with the advantage becoming pronounced at larger scales.

1 INTRODUCTION

Extreme multiclass classification, where each example is associated with exactly one label from a very large class set, arises in applications such as face recognition, landmark recognition, and species recognition, often on datasets containing hundreds of millions of examples. Cross-entropy is the standard objective for multiclass classification, but its normalization requires computing the logits of all classes, which becomes prohibitively expensive when the number of classes is large.

Practical systems therefore score each example against only a small subset of negative classes, e.g., uniformly sampled classes or in-batch classes (An et al., 2021; Dahiya et al., 2023). This makes each step cheap but provides no convergence guarantee for the objective to be minimized: placing a sampled normalizer inside the logarithm yields stochastic gradients that are biased with respect to the full cross-entropy loss. To reduce the estimation error, several methods select hard negative classes for sampled softmax, but these methods introduce additional computational burden, e.g., clustering and nearest-neighbor queries (Song et al., 2020; Zhao et al., 2021). More importantly, no choice of negative-sampling strategy, including hard-negative mining, can make the sampled-softmax gradient unbiased with respect to the full cross-entropy objective (Lin et al., 2026).

A recent line of work removes the bias at its source by reformulating cross-entropy minimization as an equivalent min-min problem, in which an auxiliary variable for each example learns its softmax normalizer (Fagan & Iyengar, 2018; Wei et al., 2026b). With the auxiliary variables held fixed, the objective is linear in the inner summation over all classes, so uniform sampling gives an unbiased

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.

gradient for the classifier at the same cost of using a sampled-softmax loss. Built on this idea, the recent work (Wei et al., 2026b) proposes an efficient algorithm SCENT, which makes the optimization provably convergent at the optimal rate for learning linear models. In practice, however, *SCENT with uniform sampling* for extreme multiclass classification is slow because its learning rate can be substantially, and in some datasets more than an order of magnitude, smaller than using full softmax. Replacing uniform samples with in-batch classes accelerates convergence, but reintroduces the bias, since in-batch classes are drawn in proportion to class frequency rather than uniformly. This raises the question studied in this paper:

How can we accelerate min-min optimization of cross-entropy loss without sacrificing unbiasedness?

We address this question by examining the source of sampling variance for learning a linear model. Inspecting the distribution of negative logits at checkpoints throughout training reveals three key patterns (Section 3). First, a handful of hard negative classes—often fewer than ten—account for nearly all of the negative contribution to each normalizer. Second, these classes are largely example-specific: even examples from the same class need not share them. Third, their identities stabilize early in training. Together, these observations suggest that a small, persistent, per-example set can capture most of the negative contribution to the normalizer.

These observations motivate PEACH, an unbiased min-min optimization method based on *personalized negative caching and sampling*. PEACH maintains a small cache of hard negative classes for each training example, evaluates their contributions exactly, and uses simple random sampling to estimate the sum over the remaining classes. The cache is updated online by ordering the logits of cached and sampled negative classes from the current iteration and retaining the highest-scoring ones for the example's next visit. This design concentrates computation on hard negatives while preserving unbiasedness and avoiding full-class scans. We summarize our key contributions below.

- Inspired by empirical observations, we propose a strategy to cache the hard negatives and sample the rest for estimating the normalizer of the cross-entropy loss for each example. We show that the resulting stochastic gradient for min-min optimization maintains unbiasedness. Based on this strategy, we propose a new method PEACH for min-min optimization of extreme multiclass classification by updating the per-example caches online using only mini-batch data.
- We prove that PEACH's gradient estimator can substantially reduce the variance compared to uniform negative sampling whenever the cache captures part of the dominant negatives, with the reduction growing as more hard negatives are evaluated exactly. Lower variance directly supports a larger learning rate that guarantees greater descent, explaining why PEACH supports faster convergence than SCENT with uniform sampling.
- We conduct experiments for learning linear models with pretrained encoders across four datasets spanning up to hundreds of thousands of classes and hundreds of millions of examples. PEACH achieves up to $\mathbf{2}$–$\mathbf{10}\times$ speedup compared with different biased and unbiased optimization methods, with the advantage becoming pronounced at larger scales.

2 Related Work

Extreme Multiclass Classification. Extreme multiclass classification dates back to Bengio & Senécal (2003; 2008), who propose stochastic approaches based on importance sampling to approximate the expensive softmax normalization. Due to the large number of classes, substantial effort has been devoted to reducing the cost of computing the logits of all classes. One line of work retains the full classification head but samples a subset of classes at each iteration, with different selection strategies (Vijayanarasimhan et al., 2015; Spring & Shrivastava, 2017; Song et al., 2020; An et al., 2021; Zhao et al., 2021). Another line of work focuses on reducing the size of the classification head, by leveraging compressed fully-connected layer (Li et al., 2021b), dynamic class pools (Wang et al., 2022; Li et al., 2021a) or hashed classifiers (Medini et al., 2019). Besides, many approaches have been explored for extreme classification in the multilabel setting, including linear one-vs-all and label-tree classifiers (Lin et al., 2024; Khandagale et al., 2020; Onishi & Hayashi, 2025), deep encoders that score a subset of labels (You et al., 2019; Chang et al., 2020; Dahiya et al., 2021b; Kharbanda et al., 2022), dual encoder architectures (Dahiya et al., 2021a; Gupta et al., 2024; Kharbanda et al., 2025; Dahiya et al., 2025; Yadav et al., 2024; Wang et al., 2025; Ortego et al., 2026), system-level

optimization (Schultheis & Babbar, 2023; Ullah et al., 2024; Zhang et al., 2025; Ullah et al., 2026). In this work, we focus on the multiclass setting and leave multilabel classification as a future direction.

Optimization of Log-Sum-Exp. Log-Sum-Exp objective arises in many areas, including extreme classification, listwise learning to rank (Xia et al., 2008), contrastive learning (Yuan et al., 2022), partial AUC optimization (Zhu et al., 2022), and KL-regularized distributionally robust optimization (Qi et al., 2021). To avoid the costly computation of the summation (e.g., summation over all classes in Extreme Classification), one naive approach is to use a mini-batch of samples for estimating the average inside the logarithmic function (Hu et al., 2020), which yields a biased gradient estimator (Lin et al., 2026). Fagan & Iyengar (2018) develop an equivalent min-min formulation of softmax with auxiliary log-normalizers to avoid the bias. NeuCLIP (Wei et al., 2026a) approximates the log-normalizers using a neural network, and Wei et al. (2026b) propose a geometry-aware update for the auxiliary variable. Compositional optimization methods are also designed to solve Log-Sum-Exp (Wang & Yang, 2022; Yuan et al., 2022), which track the inner average across iterations. Gladin et al. (2026) propose to optimize a SoftPlus approximation of the Log-Sum-Exp objective instead.

Hard Negative Mining and Persistent Caches. Hard negative mining identifies negative classes to which the model assigns high score given an input sample, and the goal is to focus training on these confusing classes. For extreme classification, KNN Softmax (Song et al., 2020) selects classes that are neighbors of the positive class in a constructed classifier-weight graph. ANN Softmax (Zhao et al., 2021) identifies hard negative classes by measuring the similarity between the input sample and class weights. NGAME (Dahiya et al., 2023) extracts hard negative labels using the distance between sample and label embeddings. Persistent cache complements hard negative mining by retaining negative samples to reduce the cost of mining. NSCaching (Zhang et al., 2019) maintains two caches to retain hard negative samples for knowledge-graph embedding, and updates the cache at every iteration. Wang et al. (2020); Lindgren et al. (2021); Chen et al. (2022) also keep track of embeddings from previous iterations so that the model has access to more hard negative samples, and update the cache at each iteration. These caching approaches focus on tasks other than extreme classification, including information retrieval, deep metric learning and knowledge-graph embedding.

3 Preliminaries and Empirical Motivation

We consider the extreme multiclass classification problem, where each example has one unique label and the number of classes C is extremely large. Given a training set of n examples with frozen features $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in [C] = \{1, \dots, C\}$, our goal is to learn a linear classifier $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_C]^\top \in \mathbb{R}^{C \times d}$. Let $\mathbf{e}_j \in \mathbb{R}^C$ denote the j-th standard basis vector. For example i and a negative class $j \neq y_i$, we define the logit difference $z_{ij}(\mathbf{W})$, the negative partition $A_i(\mathbf{W})$ and the averaged partition $g_i(\mathbf{W})$ as

$$z_{ij}(\mathbf{W}) \coloneqq \mathbf{x}_i^\top \mathbf{w}_j - \mathbf{x}_i^\top \mathbf{w}_{y_i}, \qquad A_i(\mathbf{W}) \coloneqq \sum\nolimits_{j \neq y_i} e^{z_{ij}(\mathbf{W})}, \qquad g_i(\mathbf{W}) = \frac{1 + A_i(\mathbf{W})}{M}, \quad (1)$$

where $M := C - 1$. Then the empirical cross-entropy objective can be written as:

$$\mathcal{L}(\mathbf{W}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} -\log\left(\frac{e^{\mathbf{x}_i^\top \mathbf{w}_{y_i}}}{\sum_j e^{\mathbf{x}_i^\top \mathbf{w}_j}}\right) = \frac{1}{n} \sum_{i=1}^{n} \log(g_i(\mathbf{W})) + \log M. \quad (2)$$

The challenge of updating $\mathbf{W}$ for extreme classification is that even using a mini-batch of samples per iteration is computationally expensive. Below, we elaborate on this challenge and discuss several approaches. Let $\mathcal{B}_t \subset [n]$ be a uniformly sampled mini-batch of size B, and we define $\mathbf{\Delta}_{ij} := \nabla_{\mathbf{W}} z_{ij}(\mathbf{W}) = (\mathbf{e}_j - \mathbf{e}_{y_i})\mathbf{x}_i^\top$. Then an unbiased mini-batch stochastic gradient can be computed by

$$\mathbf{G}_t^{\text{full}} = \frac{1}{B} \sum_{i \in \mathcal{B}_t} \frac{\sum_{j \neq y_i} e^{z_{ij}(\mathbf{W}_t)} \mathbf{\Delta}_{ij}}{1 + A_i(\mathbf{W}_t)}. \quad (3)$$

Throughout the paper, we refer to an SGD-based optimizer using the above stochastic gradient for updating $\mathbf{W}$ as **Full Softmax**. However, both numerator and denominator in the stochastic gradient of each example require summing over $C - 1$ classes. Thus, the complexity of computing $\mathbf{G}_t^{\text{full}}$ is $\mathcal{O}(BCd)$, making it prohibitive for extreme classification.

Biased stochastic gradients using sampled negative classes. Some heuristic approaches simply use in-batch negative classes to estimate $A_i(\mathbf{W}_t)$, which are the labels of other examples in the same mini-batch but from different classes. Specifically, for example i, we define $\mathcal{S}_{i,t}^{\mathrm{IB}} \coloneqq \{y_k : k \in \mathcal{B}_t,\ y_k \neq y_i\}$. The gradient estimator is computed by:

$$\mathbf{G}_t^{\mathrm{IB}} = \frac{1}{B} \sum\nolimits_{i \in \mathcal{B}_t} \frac{\sum_{j \in \mathcal{S}_{i,t}^{\mathrm{IB}}} e^{z_{ij}(\mathbf{W}_t)} \boldsymbol{\Delta}_{ij}}{1 + \sum_{j \in \mathcal{S}_{i,t}^{\mathrm{IB}}} e^{z_{ij}(\mathbf{W}_t)}}. \tag{4}$$

This is actually the gradient of the mini-batch Info-NCE loss exploited in Oord et al. (2018). This strategy introduces no additional negative-sampling cost, making it particularly attractive for large-scale training. We refer to an SGD-based optimizer using this biased gradient estimator as **BSGD (in-batch)**. A fundamental limitation is that replacing the full negative partition with a sampled subset yields a biased stochastic gradient: $\mathbb{E}_{\mathcal{B}_t, \mathcal{S}_{i,t}}[\mathbf{G}_t] \neq \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}_t)$ no matter what negative-sampling strategy is used to sample the negative classes (Lin et al., 2026).

To reduce the approximation error, hard-negative mining approaches have been proposed to construct a small set of negative classes for each example by finding hard negative labels that contribute more to $A_i(\mathbf{W}_t)$. For example, Song et al. (2020) proposed to use nearest-neighbor classes based on similarities of model weights:

$$\mathcal{S}_{i,t}^{\mathrm{KNN}} \coloneqq \mathrm{TopK}_{j \in [C] \setminus \{y_i\}}\ s(\mathbf{w}_{y_i,t}, \mathbf{w}_{j,t}), \tag{5}$$

where $s(\cdot, \cdot)$ denotes the similarity between classifier weights, and TopK returns the K most similar negative classes. The KNN-selected classes are further augmented with in-batch classes, and additional labels are sampled uniformly from the remaining classes until the negative set reaches $0.1C$ classes (Song et al., 2020). The resulting gradient estimator is computed analogously to (4) using this augmented negative set. We refer to an SGD-based optimizer using this biased gradient estimator as **BSGD (KNN)**. Constructing the KNN set requires additional nearest-neighbor search and therefore incurs extra computational cost.

Min-min optimization. To avoid the bias issue in stochastic optimization, we can cast the cross-entropy loss as a minimization problem over an auxiliary variable. Using $\log g = \min_{\nu \in \mathbb{R}} \{\nu + g e^{-\nu} - 1\}$ with the optimal solution ν_* satisfying $e^{\nu_*} = g$, the problem (2) is equivalent, up to the constant $\log M$, to

$$\min_{\mathbf{W}, \boldsymbol{\nu} \in \mathbb{R}^n} \Phi(\mathbf{W}, \boldsymbol{\nu}) = \frac{1}{n} \sum_{i=1}^{n} \left[\nu_i + e^{-\nu_i} g_i(\mathbf{W}) - 1 \right]. \tag{6}$$

Since the objective becomes linear in terms of summation over all data and all classes, unbiased stochastic gradient estimators of $\mathbf{W}, \boldsymbol{\nu}$ can be easily constructed. Built on this formulation, Wei et al. (2026b) proposed an efficient algorithm SCENT for optimizing the cross-entropy objective. With a uniformly sampled negative set $\mathcal{S}_{i,t} \subset [C] \setminus \{y_i\}$, $g_i(\mathbf{W}_t)$ is estimated by

$$\hat{g}_{i,t} = \frac{1}{M} + \frac{1}{|\mathcal{S}_{i,t}|} \sum\nolimits_{j \in \mathcal{S}_{i,t}} e^{z_{ij}(\mathbf{W}_t)}. \tag{7}$$

Then, SCENT updates $\nu_{i,t}$ for $i \in \mathcal{B}_t$ by stochastic proximal mirror descent method, yielding:

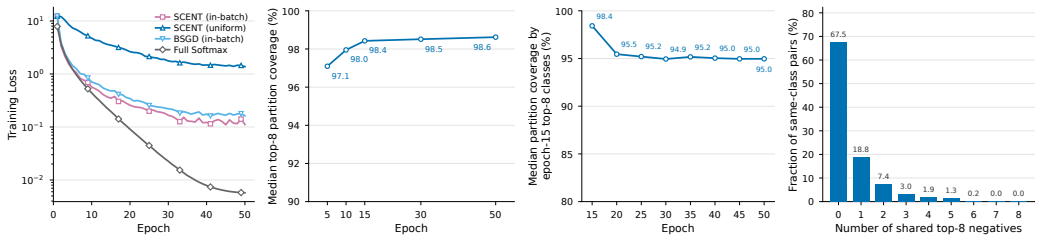
$$\nu_{i,t} = \nu_{i,t-1} + \log\left(1 + \alpha \hat{g}_{i,t}\right) - \log\left(1 + \alpha e^{\nu_{i,t-1}}\right), \tag{8}$$

and updates $\mathbf{W}_t$ by using the following gradient estimator

$$\mathbf{G}_t^{\mathrm{SCENT}} = \frac{1}{B} \sum_{i \in \mathcal{B}_t} e^{-\nu_{i,t}} \left[\frac{1}{|\mathcal{S}_{i,t}|} \sum_{j \in \mathcal{S}_{i,t}} e^{z_{ij}(\mathbf{W}_t)} \boldsymbol{\Delta}_{ij} \right], \tag{9}$$

where $\alpha > 0$ is a learning rate parameter for updating $\boldsymbol{\nu}$. Similar to BSGD, this estimator only requires a time complexity of $\mathcal{O}(Bmd)$. Under uniform negative sampling such that $\mathbb{E}[\hat{g}_{i,t} \mid \mathbf{W}_t] = g_i(\mathbf{W}_t)$, SCENT has been shown to converge to an ϵ-optimal solution with an iteration complexity in the order of $\mathcal{O}(1/\epsilon^2)$.

While enjoying a convergence guarantee, SCENT with uniform negative class sampling converges slowly in practice, as shown in Fig. 1a. In experiments of Wei et al. (2026b), the authors instead use in-batch negative classes in (7), i.e., $\mathcal{S}_{i,t} = \mathcal{S}_{i,t}^{\mathrm{IB}}$. However, the distribution of in-batch negatives is induced by the label distribution of the mini-batch rather than being uniform over all classes. Consequently, the resulting gradient estimators become biased, which will break the convergence guarantee.



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

Figure 1: Empirical observations on Google Landmarks v2. (a) Training loss across optimization methods. (b) The median values of the sum of top eight $q_{ij}^{(t)}$ across 1000 examples at different epochs. (c) For 1000 randomly selected examples, we identify their top eight hard-negative classes at epoch 15 and keep these classes fixed. Their contribution to the negative partition remains around 95% through epoch 50, showing that hard negatives may stabilize early. (d) Among pairs of examples with the same label, around 70% share none of their top-eight hard-negative classes, showing that hard negatives could be example-specific.

Empirical Motivation. If per-iteration computational cost is ignored, Full Softmax provides a natural reference for optimization efficiency in terms of iteration complexity: it evaluates the exact normalization and therefore introduces no additional randomness from negative class sampling. Prior analysis of SCENT (Wei et al., 2026b) establishes a convergence rate comparable to that of Full Softmax. In extreme classification, however, we observe that SCENT with uniform negative class sampling requires substantially more epochs to converge than Full Softmax, and can even lag behind BSGD (in-batch) methods (Fig. 1a). This gap suggests that unbiasedness alone is insufficient for fast optimization and motivates us to examine the structure of the negative terms being estimated stochastically.

For each example i, we inspect the distribution of negative logit contributions at several checkpoints during training of using Full Softmax. Given the logit difference z_{ij} for negative class j, define its normalized contribution to the negative partition as

$$q_{ij}^{(t)} = \frac{e^{z_{ij}(\mathbf{W}_t)}}{A_i(\mathbf{W}_t)}, \qquad \sum\nolimits_{j \neq y_i} q_{ij}^{(t)} = 1. \tag{10}$$

Thus, $q_{ij}^{(t)}$ measures the fraction of the negative partition $A_i(\mathbf{W}_t)$ contributed by class j. High values of $q_{ij}^{(t)}$ correspond to hard negative classes. Examining the distribution of $\{q_{ij}^{(t)}\}_{j \neq y_i}$ across epochs reveals three consistent patterns. Below, we only report results on the Google Landmarks v2 dataset ($N = 4{,}132{,}914, C = 203{,}094$), but similar trends occur in other datasets investigated in Section 6.

The negative partition is highly concentrated. For each example, only a small number of negative classes contribute substantially to $A_i(\mathbf{W}_t)$. As shown in Fig. 1b, the top eight negatives already account for most of the negative partition throughout training. Consequently, although the label space may contain hundreds of thousands of classes, most negative classes make only negligible contributions for a given example.

Hard negatives stabilize early in training. The largest-contributing negatives change slowly over training. Fig. 1c shows that negatives identified at an early checkpoint continue to account for most of the negative partition at later epochs. This persistence makes it possible to reuse previously identified negatives across repeated visits to the same example.

Hard negatives can be example-specific. The identities of the largest-contributing negative classes need not be shared across examples with the same ground-truth label. As shown in Fig. 1d, same-class example pairs often have little overlap between their top negative sets. This suggests that class-level sharing alone may be insufficient to represent the negative structure relevant to each individual example.

Together, these observations reveal a simple empirical structure: for each instance, the negative partition is concentrated on a small, persistent, and potentially example-specific set of classes. This motivates evaluating these classes deterministically while estimating the contribution of the remaining classes by random sampling.

Algorithm 1 PEACH

Initialize classifier $\mathbf{W}$, auxiliary variables $\boldsymbol{\nu}$, caches $\{\mathcal{H}_i\}_{i=1}^n$
Initialize cache size L, uncached budget m, learning rates η and α
for $t = 1, \ldots, T$ **do**
  Sample a mini-batch $\mathcal{B}_t$
  for each $i \in \mathcal{B}_t$ **do**
    Read the cache $\mathcal{H}_{i,t}$
    Sample $\mathcal{S}_{i,t}^r \subset [C] \setminus (\mathcal{H}_{i,t} \cup \{y_i\})$ uniformly without replacement, with $|\mathcal{S}_{i,t}^r| = m$
    Compute $\tilde{A}_i(\mathbf{W}_t)$ according to (11) and $\tilde{g}_i(\mathbf{W}_t) = \frac{1+\tilde{A}_i(\mathbf{W}_t)}{M}$
    Update $\nu_{i,t} = \nu_{i,t-1} + \log\left(1 + \alpha \tilde{g}_i(\mathbf{W}_t)\right) - \log\left(1 + \alpha e^{\nu_{i,t-1}}\right)$
  end for
  Compute gradients $\mathbf{G}_t = \frac{1}{B} \sum_{i \in \mathcal{B}_t} e^{-\nu_{i,t}} \nabla \tilde{g}_i(\mathbf{W}_t)$.
  Update $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta \mathbf{G}_t$ ◇ (momentum-based update can be used in practice)
  for each $i \in \mathcal{B}_t$ **do**
    Collect negative classes observed at iteration t
    $\mathcal{H}_{i,t+1} \leftarrow$ TOP-L candidates according to (12)
  end for
  For each $i \notin \mathcal{B}_t$, $\nu_{i,t} = \nu_{i,t-1}$ and $\mathcal{H}_{i,t+1} = \mathcal{H}_{i,t}$
end for

4 PEACH: CACHE THE HARD AND SAMPLE THE REST

Motivated by the observation in Section 3, we propose PEACH to accelerate optimization of the min–min reformulation of the cross-entropy objective. The key idea is to maintain and update a small cache of hard negative class indices for each example. We decompose the sum over all negative classes in $A_i(\mathbf{W}_t)$ into two parts: the contribution of cached classes, which is evaluated exactly, and that of the remaining classes, which is estimated via uniform sampling.

For example i, let $\mathcal{H}_{i,t}$ denote its cached hard negative classes at the t-th iteration, where $|\mathcal{H}_{i,t}| = L$. We draw m negatives uniformly from the uncached negative label set $[C] \setminus (\mathcal{H}_{i,t} \cup \{y_i\})$ denoted by $\mathcal{S}_{i,t}^r$. Then, we estimate the negative partition by

$$\tilde{A}_i(\mathbf{W}_t) = \sum_{j \in \mathcal{H}_{i,t}} e^{z_{ij}(\mathbf{W}_t)} + \frac{M - L}{m} \sum_{j \in \mathcal{S}_{i,t}^r} e^{z_{ij}(\mathbf{W}_t)}. \tag{11}$$

Conditioned on the cache snapshot $\mathcal{H}_{i,t}$ and $\mathbf{W}_t$, we have $\mathbb{E}[\tilde{A}_i(\mathbf{W}_t) \mid \mathcal{H}_{i,t}, \mathbf{W}_t] = A_i(\mathbf{W}_t)$. As a result, an unbiased estimator of $g_i(\mathbf{W}_t)$ can be computed by:

$$\tilde{g}_i(\mathbf{W}_t) = \frac{1 + \tilde{A}_i(\mathbf{W}_t)}{M}$$

Then, we plug this estimator into the SCENT algorithm for updating both $\boldsymbol{\nu}_t$ and $\mathbf{W}_t$.

This construction differs fundamentally from existing hard-negative sampling (Song et al., 2020; Zhao et al., 2021). Rather than replacing the full softmax normalization with a selected subset of hard classes, PEACH evaluates the contributions of cached classes exactly and uses unbiased sampling only for estimating that of the uncached classes. Equation (11) ensures that combining the exact contribution of cached hard negatives with the sampled estimate of the remaining negatives yields an unbiased estimator of the negative partition. Thus, the cache reallocates computation across classes while preserving unbiasedness.

A remaining question is how to compute $\mathcal{H}_{i,t}$. Unlike prior studies that use expensive hard negative mining approaches, we update $\mathcal{H}_{i,t}$ in an online fashion inspired by our empirical observations. After updating the classifier, PEACH refreshes the cache using negative classes observed during the iteration, where the logits are already computed at $\mathbf{W}_t$. Specifically, for each $i \in \mathcal{B}_t$, we form the candidate pool as the union of current cache set and the sampled uncached negative set at iteration t: $\mathcal{C}_{i,t} = \mathcal{H}_{i,t} \cup \mathcal{S}_{i,t}^r$. We retain the L candidates with the largest logit differences as $\mathcal{H}_{i,t}$, i.e.,

$$\mathcal{H}_{i,t+1} = \mathrm{TopL}_{j \in \mathcal{C}_{i,t}} z_{ij}(\mathbf{W}_t). \tag{12}$$

For examples not visited at iteration t, the cache remains unchanged.

5 WHY PEACH ACCELERATES MIN-MIN OPTIMIZATION

Below, we provide analysis on the variance reduction effect of PEACH's gradient estimator in comparison with SCENT with uniform sampling for optimizing the objective $\Phi(\mathbf{W}, \boldsymbol{\nu})$ in (6). To this end, we will consider a fixed ν_i and analyze the variance of stochastic gradient estimator of $\mathbf{W}$. When no confusion arises, we abbreviate $z_{ij}(\mathbf{W})$ as z_{ij}. The following analysis applies to any iteration t, and hence the dependence on t is omitted.

First, we analyze the variance of the gradient estimator based on uniform sampling. For example i, with a uniformly sampled negative set $\mathcal{S}_i$ of size m, the full gradient and the sampled gradient estimator of SCENT are:

$$\mathbf{G}_i = \frac{e^{-\nu_i}}{M} \sum_{j \neq y_i} e^{z_{ij}} \boldsymbol{\Delta}_{ij}, \qquad \widehat{\mathbf{G}}_i = \frac{e^{-\nu_i}}{m} \sum_{j \in \mathcal{S}_i} e^{z_{ij}} \boldsymbol{\Delta}_{ij}. \tag{13}$$

We define the relative gradient variance as

$$\rho_i := \frac{\mathbb{E}\Big[\|\widehat{\mathbf{G}}_i - \mathbf{G}_i\|_F^2\Big]}{\|\mathbf{G}_i\|_F^2}. \tag{14}$$

Besides, we define $\pi_i = \sum_{j \neq y_i} q_{ij}^2$ and $\kappa_i = \pi_i M$. The concentration measure $\pi_i = \sum_{j \neq y_i} q_{ij}^2$ characterizes how unevenly the negative partition is distributed across classes. If all negatives contribute equally, then $\pi_i = 1/M$ and $\kappa_i = 1$, whereas concentration on a single class gives $\pi_i = 1$ and $\kappa_i = M$. More generally, the empirical observation that only a few negatives dominate the partition implies $\pi_i \gg 1/M$ and hence a large κ_i.

Proposition 1 (Variance of Uniform Sampled Estimator). *If $\mathcal{S}_i \subset [C] \setminus \{y_i\}$ is a uniformly sampled subset of m negative classes without replacement, then*

$$\rho_i^{\text{unif}} = \frac{M - m}{M - 1} \frac{2\kappa_i - 1 - \pi_i}{m(1 + \pi_i)}. \tag{15}$$

Since $m \ll M$, Proposition 1 shows that ρ_i^{unif} scales approximately with κ_i/m. In particular, when the negative partition is concentrated on $\mathcal{O}(1)$ classes, $\pi_i = \Theta(1)$ and $\rho_i^{\text{unif}} = \Theta(C/m)$. This result indicates that SCENT with uniform negative sampling has high relative sampling variance in extreme classification.

Next, we analyze the variance of the gradient estimator used by PEACH. For example i, let $\mathcal{H}_i$ be the hard negative cache of size L and $\mathcal{S}_i^r$ be a sampled set of size m from the uncached negative set. Define[1]

$$A_i^{\text{unc}} = \sum_{j \notin \mathcal{H}_i,\, j \neq y_i} e^{z_{ij}}, \qquad \pi_i^{\text{unc}} = \sum_{j \notin \mathcal{H}_i,\, j \neq y_i} \left(\frac{e^{z_{ij}}}{A_i^{\text{unc}}}\right)^2, \qquad \kappa_i^{\text{unc}} = (M - L)\pi_i^{\text{unc}}. \tag{16}$$

The following proposition establishes the relative variance of PEACH's gradient estimator:

$$\widetilde{\mathbf{G}}_i = e^{-\nu_i} \nabla \tilde{g}_i(\mathbf{W}) = \frac{e^{-\nu_i}}{M} \left(\sum_{j \in \mathcal{H}_i} e^{z_{ij}} \Delta_{ij} + \frac{M - L}{m} \sum_{j \in \mathcal{S}_i^r} e^{z_{ij}} \Delta_{ij} \right). \tag{17}$$

Proposition 2 (Variance of PEACH's gradient estimator). *Conditioned on a cache $\mathcal{H}_i$, if $\mathcal{S}_i^r$ is a uniformly sampled subset of m negatives from the uncached negatives set without replacement, PEACH's gradient estimator is unbiased and has relative sampling variance*

$$\rho_i^{\text{PEACH}} = \frac{R - m}{R - 1} \left(\frac{A_i^{\text{unc}}}{A_i}\right)^2 \frac{2\kappa_i^{\text{unc}} - 1 - \pi_i^{\text{unc}}}{m(1 + \pi_i)}, \tag{18}$$

where $R = M - L$.

[1] Here, 'unc' represents 'uncached'

Proposition 2 implies that caching reduces variance through two effects. First, evaluating cached negatives deterministically leaves only an A_i^{unc}/A_i fraction of the negative partition to be estimated stochastically, yielding the squared reduction factor $(A_i^{\mathrm{unc}}/A_i)^2$. Second, the remaining variance depends on the concentration of the uncached contributions through π_i^{unc}.

When $L, m \ll C$ and the negative contributions are sufficiently concentrated, the variance ratio simplifies to (derivation shown in Appendix E.5)

$$\frac{\rho_i^{\mathrm{PEACH}}}{\rho_i^{\mathrm{unif}}} = \mathcal{O}\left(\left(\frac{A_i^{\mathrm{unc}}}{A_i}\right)^2 \frac{\pi_i^{\mathrm{unc}}}{\pi_i}\right). \tag{19}$$

Thus, caching hard negatives can substantially reduce sampling variance by removing a large portion of the negative partition from stochastic estimation.

Finally, we note that a smaller per-iteration gradient variance supports a larger learning rate and hence a greater descent (see Appendix D). We provide more details of analysis in Appendix E.

6 Experiments

Datasets. We evaluate PEACH on four large-scale multiclass classification datasets: TreeOfLife-10M ($N = 9{,}533{,}174, C = 161{,}463$) (Stevens et al., 2024), Google Landmarks v2 ($N = 4{,}132{,}914, C = 203{,}094$) (Weyand et al., 2020), Glint360k ($N = 17{,}091{,}657, C = 360{,}232$) (An et al., 2021), and TreeOfLife-200M ($N = 209{,}134{,}719, C = 795{,}533$) (Gu et al., 2026). For each dataset, we use an $80\%/10\%/10\%$ train/validation/test split and ensure that every class appearing in the validation or testing set is represented in the training set. We use frozen pretrained features throughout and optimize only the linear classifier. Additional dataset statistics are provided in Appendix B.

Baselines. We compare PEACH with five methods: Full Softmax, BSGD (in-batch), BSGD (KNN), SCENT (uniform), and SCENT (in-batch). These methods are introduced in Section 3.

Implementation. PEACH maintains one auxiliary variable ν_i and $L = 8$ cached class identifiers for each example. These per-example states are stored in CPU memory and therefore do not increase the persistent GPU-memory footprint. We use classifier-parallel distributed training on four NVIDIA A100 GPUs, with the classifier rows partitioned approximately evenly across GPUs so that each worker stores and updates only its assigned class weights. To preserve the computational benefit of negative class sampling, we further use a sparse momentum optimizer that updates only the classifier rows involved in the current iteration. For all experiments in this section, we run each method three times with different random seeds. Additional implementation details are provided in Appendix C.

6.1 Optimization Efficiency and Test Accuracy

The top row of Figure 2 compares training loss with respect to wall-clock time when all methods are run with 50 epochs. Across all datasets, PEACH reaches lower training loss within the same time budget than the existing methods. In particular, its efficiency advantage over BSGD (KNN) becomes more pronounced on larger-scale datasets, where nearest-neighbor retrieval over a large label space incurs increasing computational overhead. The bottom row shows the corresponding epoch-wise convergence. PEACH consistently converges faster than SCENT (uniform), SCENT (in-batch), BSGD (in-batch), and BSGD (KNN), while remaining competitive with Full Softmax. Overall, PEACH achieves faster optimization with better practical training efficiency, especially at larger label scales. At the same time budget, Table 1 reports test accuracy using the checkpoint with the best validation accuracy for each method. PEACH achieves the best test accuracy across all four datasets, while its advantage is pronounced on the TreeOfLife-200M over existing methods.

6.2 Relative Sampling Variance Examination

We next empirically examine the variance-reduction mechanism analyzed in Section 5. As shown in Figures 3a and 3b, PEACH consistently exhibits lower relative sampling variance ρ_i than SCENT (uniform), with the gap generally widening during training. We further evaluate cache quality by evaluating the uncached partition ratio A_i^{unc}/A_i. As shown in Figures 3c and 3d, this ratio decreases during training and approaches that computed by removing the exact top-8 negatives at each epoch,

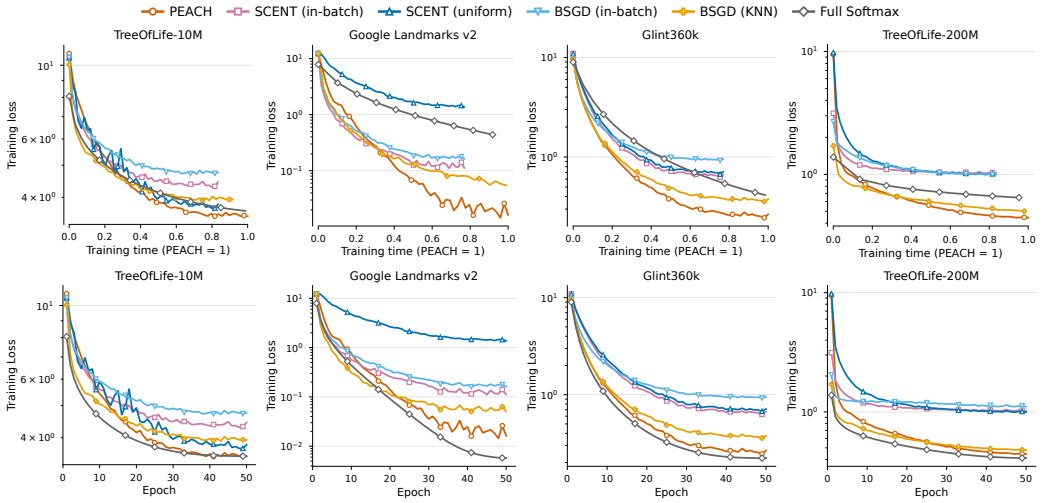


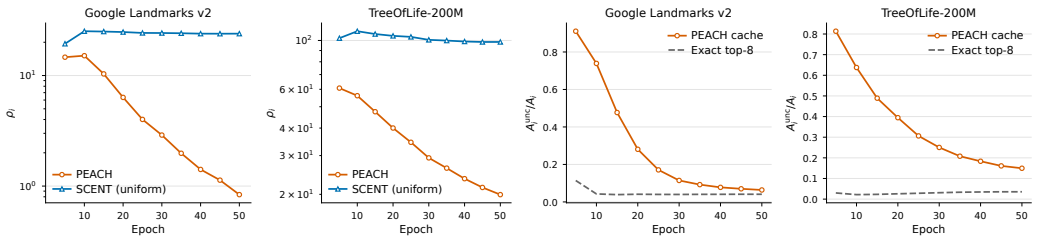
Figure 2: Training loss across the four datasets. Top: loss versus normalized wall-clock time, where the total running time of PEACH is normalized to one. Bottom: loss versus epoch.

Table 1: Test accuracy (%) of different methods with the same compute budget across four datasets. Results are reported as mean (std).

Method	TreeOfLife-10M	Google Landmarks v2	Glint360k	TreeOfLife-200M
Training Hours	0.16	0.32	1.07	8.00
Full Softmax	45.03 (0.01)	50.02 (0.00)	**99.93 (0.00)**	82.99 (0.00)
BSGD (in-batch)	43.51 (0.03)	49.62 (0.01)	99.90 (0.00)	77.47 (0.04)
SCENT (in-batch)	43.53 (0.03)	50.10 (0.03)	99.90 (0.01)	78.53 (0.03)
SCENT (uniform)	45.20 (0.01)	46.98 (0.01)	99.89 (0.00)	80.36 (0.02)
BSGD (KNN)	45.60 (0.02)	50.53 (0.01)	**99.93 (0.00)**	83.95 (0.04)
PEACH	**45.65 (0.01)**	**50.65 (0.01)**	**99.93 (0.00)**	**84.60 (0.02)**

indicating that the online cache progressively captures high-contribution negatives. Together, these results support the variance-reduction mechanism characterized in Equation (19).

Finally, we report ablations on the cache size L and comparison with BSGD with per-example cache and SCENT with KNN in Appendix F.



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

Figure 3: (a)(b): Mean relative sampling variance ρ_i of PEACH and SCENT (uniform) during training. (c)(d): Median uncached partition ratio A_i^{unc}/A_i for PEACH cache and exact top-8 negatives.

7 Conclusion

We introduced PEACH, an efficient and unbiased method for extreme multiclass classification using personalized negative caching and sampling. PEACH exploits the concentration of the negative partition on a small set of persistent, example-specific hard negatives, evaluating them exactly while sampling the rest. Our analysis shows reduced sampling variance while preserving unbiasedness, and experiments on large-scale datasets demonstrate faster optimization and better predictive performance.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, manuscript organization, and parts of the code implementation. All AI-assisted text and code were reviewed, tested, and verified by the authors. The authors take responsibility for the mathematical statements, experimental results, citations, technical claims, and implementation in the paper.

REFERENCES

Xiang An, Xuhan Zhu, Yuan Gao, Yang Xiao, Yongle Zhao, Ziyong Feng, Lan Wu, Bin Qin, Ming Zhang, Debing Zhang, and Ying Fu. Partial fc: Training 10 million identities on a single machine. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 1445–1449, 2021. doi: 10.1109/ICCVW54120.2021.00166.

Yoshua Bengio and Jean-Sébastien Senécal. Quick training of probabilistic neural nets by importance sampling. In *International Workshop on Artificial Intelligence and Statistics*, pp. 17–24. PMLR, 2003.

Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713, 2008.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3163–3171, 2020.

Jin Chen, Defu Lian, Yucheng Li, Baoyun Wang, Kai Zheng, and Enhong Chen. Cache-augmented inbatch importance resampling for training recommender retriever. *Advances in Neural Information Processing Systems*, 35:34817–34830, 2022.

Kunal Dahiya, Ananye Agarwal, Deepak Saini, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, Manik Varma, et al. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International conference on machine learning*, pp. 2330–2340. PMLR, 2021a.

Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 31–39, 2021b.

Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Gururaj K, Prasenjit Dey, Amit Singh, et al. Ngame: Negative mining-aware mini-batching for extreme classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 258–266, 2023.

Kunal Dahiya, Diego Ortego, and David Jimenez-Cabello. Prototypical extreme multi-label classification with a dynamic margin loss. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10709–10727, 2025.

Francois Fagan and Garud Iyengar. Unbiased scalable softmax optimization. *arXiv preprint arXiv:1803.08577*, 2018.

Egor Gladin, Alexey Kroshnin, Jia-Jie Zhu, and Pavel Dvurechensky. Improved stochastic optimization of logsumexp. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=TzQElzflxR.

Jianyang Gu, Sam Stevens, Elizabeth Campolongo, Matthew Thompson, Net Zhang, Jiaman Wu, Andrei Kopanev, Zheda Mai, Alexander White, James Balhoff, et al. Bioclip 2: Emergent properties from scaling hierarchical contrastive learning. *Advances in Neural Information Processing Systems*, 38:102778–102811, 2026.

Nilesh Gupta, Fnu Devvrit, Ankit Singh Rawat, Srinadh Bhojanapalli, Prateek Jain, and Inderjit Dhillon. Dual-encoders for extreme multi-label classification. In *International Conference on Learning Representations*, volume 2024, pp. 52858–52884, 2024.

Yifan Hu, Siqi Zhang, Xin Chen, and Niao He. Biased stochastic first-order methods for conditional stochastic optimization and applications in meta learning. *Advances in Neural Information Processing Systems*, 33:2759–2770, 2020.

Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109(11):2099–2119, 2020.

Siddhant Kharbanda, Atmadeep Banerjee, Erik Schultheis, and Rohit Babbar. Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. *Advances in neural information processing systems*, 35:2074–2087, 2022.

Siddhant Kharbanda, Devaansh Gupta, Gururaj K, Pankaj Malhotra, Amit Singh, Cho-Jui Hsieh, and Rohit Babbar. Unidec: Unified dual encoder and classifier training for extreme multi-label classification. In *Proceedings of the ACM on Web Conference 2025*, pp. 4124–4133, 2025.

Bi Li, Teng Xi, Gang Zhang, Haocheng Feng, Junyu Han, Jingtuo Liu, Errui Ding, and Wenyu Liu. Dynamic class queue for large scale face recognition in the wild. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3762–3771. IEEE, 2021a.

Pengyu Li, Biao Wang, and Lei Zhang. Virtual fully-connected layer: Training a large-scale face recognition dataset with limited computational resources. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13315–13324, June 2021b.

He-Zhe Lin, Cheng-Hung Liu, and Chih-Jen Lin. Exploring space efficiency in a tree-based linear model for extreme multi-label classification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16245–16260, 2024.

Li-Chung Lin, Yaxu Liu, and Chih-Jen Lin. Sampled estimators for softmax must be biased. *Advances in Neural Information Processing Systems*, 38:46757–46780, 2026.

Erik Lindgren, Sashank Reddi, Ruiqi Guo, and Sanjiv Kumar. Efficient training of retrieval models using negative cache. *Advances in Neural Information Processing Systems*, 34:4134–4146, 2021.

Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. *Advances in Neural Information Processing Systems*, 32, 2019.

Kazuma Onishi and Katsuhiko Hayashi. A simple but effective closed-form solution for extreme multi-label learning. In *European Conference on Information Retrieval*, pp. 265–274. Springer, 2025.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Diego Ortego, Marlon Rodríguez, Mario Almagro, Kunal Dahiya, David Jiménez, and Juan C SanMiguel. Large language models meet extreme multi-label classification: scaling and multi-modal framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24630–24638, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/39647.

Qi Qi, Zhishuai Guo, Yi Xu, Rong Jin, and Tianbao Yang. An online method for a class of distributionally robust optimization with non-convex objectives. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 10067–10080. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/533fa796b43291fc61a9e812a50c3fb6-Paper.pdf.

Erik Schultheis and Rohit Babbar. Towards memory-efficient training for extremely large output spaces–learning with 670k labels on a single commodity gpu. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 689–704. Springer, 2023.

Liuyihan Song, Pan Pan, Kang Zhao, Hao Yang, Yiming Chen, Yingya Zhang, Yinghui Xu, and Rong Jin. Large-scale training system for 100-million classification at alibaba. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 2909–2930, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403342. URL https://doi.org/10.1145/3394486.3403342.

Ryan Spring and Anshumali Shrivastava. Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 445–454, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098035. URL https://doi.org/10.1145/3097983.3098035.

Samuel Stevens, Jiaman Wu, Matthew J Thompson, Elizabeth G Campolongo, Chan Hee Song, David Edward Carlyn, Li Dong, Wasila M Dahdul, Charles Stewart, Tanya Berger-Wolf, Wei-Lun Chao, and Yu Su. Bioclip: A vision foundation model for the tree of life. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19412–19424, 2024. doi: 10.1109/CVPR52733.2024.01836.

Nasib Ullah, Erik Schultheis, Mike Lasby, Yani Ioannou, and Rohit Babbar. Navigating extremes: Dynamic sparsity in large output spaces. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 117210–117236. Curran Associates, Inc., 2024. doi: 10.52202/079017-3721. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d4bdeed749a437de2cbe2e2c7e5a6a8a-Paper-Conference.pdf.

Nasib Ullah, Jinbin Zhang, Jean Lucien Randrianantenaina, Erik Schultheis, and Rohit Babbar. HASTE: Hardware-aware dynamic sparse training for large output spaces. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=PAOhstVuIt.

Sudheendra Vijayanarasimhan, Jonathon Shlens, Rajat Monga, and Jay Yagnik. Deep networks with large output spaces. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.7479.

Bokun Wang and Tianbao Yang. Finite-sum coupled compositional stochastic optimization: Theory and applications. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23292–23317. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wang22ak.html.

Kai Wang, Shuo Wang, Panpan Zhang, Zhipeng Zhou, Zheng Zhu, Xiaobo Wang, Xiaojiang Peng, Baigui Sun, Hao Li, and Yang You. An efficient training approach for very large scale face recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4073–4082, 2022. doi: 10.1109/CVPR52688.2022.00405.

Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R. Scott. Cross-batch memory for embedding learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6387–6396, 2020. doi: 10.1109/CVPR42600.2020.00642.

Yau-Shian Wang, Wei-Cheng Chang, Jyun-Yu Jiang, Jiong Zhang, Hsiang-Fu Yu, and SVN Vishwanathan. Retrieval-augmented encoders for extreme multi-label text classification. *arXiv preprint arXiv:2502.10615*, 2025.

Xiyuan Wei, Chih-Jen Lin, and Tianbao Yang. Neuclip: Efficient large-scale clip training with neural normalizer optimization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 22966–22996, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/275c17cd2729e2ef4e7a5b37b6d9b86c-Paper-Conference.pdf.

Xiyuan Wei, Linli Zhou, Bokun Wang, Chih-Jen Lin, and Tianbao Yang. A geometry-aware efficient algorithm for compositional entropic risk minimization. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=0SGle5hjIf.

Tobias Weyand, André Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2 – a large-scale benchmark for instance-level recognition and retrieval. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2572–2581, 2020. doi: 10.1109/CVPR42600.2020.00265.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1192–1199, New York, NY, USA, 2008. Association for Computing Machinery. doi: 10.1145/1390156.1390306. URL https://doi.org/10.1145/1390156.1390306.

Sachin Yadav, Deepak Saini, Anirudh Buvanesh, Bhawna Paliwal, Kunal Dahiya, Siddarth Asokan, Yashoteja Prabhu, Jian Jiao, and Manik Varma. Extreme meta-classification for large-scale zero-shot retrieval. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 3657–3666, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3672046. URL https://doi.org/10.1145/3637528.3672046.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/9e6a921fbc428b5638b3986e365d4f21-Paper.pdf.

Zhuoning Yuan, Yuexin Wu, Zi-Hao Qiu, Xianzhi Du, Lijun Zhang, Denny Zhou, and Tianbao Yang. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25760–25782. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/yuan22b.html.

Jinbin Zhang, Nasib Ullah, Erik Schultheis, and Rohit Babbar. ELMO : Efficiency via low-precision and peak memory optimization in large output spaces. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76159–76174. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhang25bv.html.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 614–625, 2019. doi: 10.1109/ICDE.2019.00061.

Kang Zhao, Liuyihan Song, Yingya Zhang, Pan Pan, Yinghui Xu, and Rong Jin. Ann softmax: acceleration of extreme classification training. *Proc. VLDB Endow.*, 15(1):1–10, September 2021. ISSN 2150-8097. doi: 10.14778/3485450.3485451. URL https://doi.org/10.14778/3485450.3485451.

Dixian Zhu, Gang Li, Bokun Wang, Xiaodong Wu, and Tianbao Yang. When AUC meets DRO: Optimizing partial AUC for deep learning with non-convex convergence guarantee. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27548–27573. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhu22g.html.

A NOTATION

Table 2: Summary of notations.

Symbol	Definition
n	Number of training examples
$[C]$	Label set, $\{1, \dots, C\}$
C	Number of classes
M	Number of negative classes, $M := C - 1$
$\mathbf{x}_i \in \mathbb{R}^d$	Frozen feature of example i
$y_i \in [C]$	Ground-truth label of example i
$\mathbf{W} \in \mathbb{R}^{C \times d}$	Linear classifier. The j-th row, $\mathbf{w}_j$, denotes the classifier for class j
z_{ij}	Logit difference, $\mathbf{x}_i^\top \mathbf{w}_j - \mathbf{x}_i^\top \mathbf{w}_{y_i}$
A_i	Negative partition, $\sum_{j \neq y_i} e^{z_{ij}}$
$\boldsymbol{\Delta}_{ij}$	Logit difference gradient, $\nabla_{\mathbf{W}} z_{ij}$
q_{ij}	Normalized contribution to the negative partition, $e^{z_{ij}}/A_i$
B	Mini-batch size
$\mathcal{B}_t$	Mini-batch data at iteration t; $\lvert\mathcal{B}_t\rvert = B$
$\mathcal{S}_{i,t}$	Sampled negative set for example i at iteration t
m	Number of sampled negative classes
ν_i	Auxiliary variable associated with example i
$\boldsymbol{\nu}$	Vector of per-example auxiliary variables
η, α	Learning rates for $\mathbf{W}$ and $\boldsymbol{\nu}$, respectively
$\mathcal{H}_{i,t}$	Cached negative classes for example i at iteration t; $\mathcal{H}_i$ when t is omitted
$\mathcal{S}^r_{i,t}$	Sampled negatives from the uncached negative set for example i
L	Cache size
$\mathcal{R}_i$	Uncached negative classes, $[C] \setminus (\mathcal{H}_i \cup \{y_i\})$
R	Number of uncached negative classes, $R := M - L$
A_i^{unc}	Uncached negative partition, $\sum_{j \notin \mathcal{H}_i,\, j \neq y_i} e^{z_{ij}}$
π_i	Concentration measure, $\sum_{j \neq y_i} q_{ij}^2$
κ_i	Scaled concentration measure, $\kappa_i = M\pi_i$
π_i^{unc}	Concentration measure over the uncached negative set
κ_i^{unc}	Scaled uncached concentration measure, $\kappa_i^{\text{unc}} = R\pi_i^{\text{unc}}$
ρ_i	Relative sampling variance of a per-example gradient estimator

B DATASET DETAILS

In Table 3 we present more details of each dataset we use for the experiments.

Table 3: Details of the datasets used in our experiments. Dim denotes the output feature dimension of the corresponding pretrained encoder.

Dataset	**Examples**	**Classes**	**Description**	**Pretrained Encoder**	**Dim**
TreeOfLife-10M	9,533,174	161,463	Biological taxon classification	BioCLIP	512
Google Landmarks v2	4,132,914	203,094	Landmark instance classification	DINOv2-L	1024
Glint360k	17,091,657	360,232	Face identity classification	ArcFace ResNet-50	512
TreeOfLife-200M	209,134,719	795,533	Biological taxon classification	BioCLIP 2	768

C Implementation Details of PEACH

C.1 Sampling

To improve sampling efficiency, we do not sample the uncached classes for each instance independently in implementation. In particular, at iteration t, we draw a single shared class set $\mathcal{S}_t^{\mathrm{sh}} \subset [C]$ of size m uniformly without replacement and reuse it for all examples in the mini-batch. For example i, the sampled uncached classes are

$$\mathcal{S}_{i,t}^{\mathrm{sh}} = \mathcal{S}_t^{\mathrm{sh}} \setminus (\mathcal{H}_{i,t} \cup \{y_i\}).$$

Since each class in $[C]$ is included in $\mathcal{S}_t^{\mathrm{sh}}$ with probability m/C, we estimate the negative partition as

$$\widehat{A}_{i,t}^{\mathrm{shared}} = \sum_{j \in \mathcal{H}_{i,t}} e^{z_{ij}} + \frac{C}{m} \sum_{j \in \mathcal{S}_{i,t}^{\mathrm{sh}}} e^{z_{ij}}.$$

Conditioned on the frozen cache, $\mathbb{E}[\widehat{A}_{i,t}^{\mathrm{shared}}] = A_i$. Thus, sharing the sampled class set across examples does not introduce bias; it only induces correlation among their sampling error. Accordingly, for fixed $\nu_{i,t}$, the shared-sampling primal estimator is

$$\widehat{\mathbf{G}}_{i,t}^{\mathrm{shared}} = \frac{e^{-\nu_{i,t}}}{M} \left[\sum_{j \in \mathcal{H}_{i,t}} e^{z_{ij}(\mathbf{W}_t)} \boldsymbol{\Delta}_{ij} + \frac{C}{m} \sum_{j \in \mathcal{S}_{i,t}^{\mathrm{sh}}} e^{z_{ij}(\mathbf{W}_t)} \boldsymbol{\Delta}_{ij} \right]. \tag{20}$$

Conditioned on the frozen cache and fixed auxiliary variable, this estimator is still unbiased for the full-class primal gradient.

C.2 Distributed Training

In extreme classification, the memory required by the classification head scales linearly with the number of classes, i.e., $\mathcal{O}(Cd)$ for a d-dimensional classifier. While this cost is manageable for datasets with a small number of classes, it quickly becomes prohibitive as the label space grows. For example, TreeOfLife-200M (Gu et al., 2026) contains 795,533 classes and the frozen features have dimension $d = 768$. Then the classification head has approximately 6.11×10^8 parameters and requires 2.44 GB memory for FP32 weights alone, before taking gradients and optimizer states for consideration. For industrial-scale extreme-classification systems with up to 10^8 classes (Song et al., 2020), the same $d = 768$ classifier would require approximately 307 GB for FP32 weights alone. This linear growth motivates sharding the classifier across devices rather than replicating the entire classification head on every GPU.

To support such large label spaces, we adopt a classifier-parallel distributed implementation commonly used in large-scale classification systems (Zhao et al., 2021; Song et al., 2020; Gupta et al., 2024). The rows of the classifier matrix $\mathbf{W}$ are partitioned approximately evenly across GPUs, so that each GPU stores and updates only a subset of class weights and computes logits only for its locally assigned classes. Consequently, the classifier memory per GPU decreases approximately in proportion to the number of GPUs, enabling PEACH to scale up to substantially larger industrial-scale classification problems.

C.3 Sparse Momentum Update

Although negative sampling avoids evaluating all classifier rows in the forward pass, a standard momentum optimizer can still induce $\mathcal{O}(Cd)$ operations during parameter updates. Let $\mathcal{A}_t \subseteq [C]$ denote the set of classifier rows that participate in the logit computation at iteration t. For a classifier row $j \notin \mathcal{A}_t$, the current gradient is zero. Nevertheless, standard SGD with momentum continues to decay its momentum buffer and updates the corresponding weight according to

$$\mathbf{b}_j \leftarrow \beta \mathbf{b}_j, \qquad \mathbf{w}_j \leftarrow \mathbf{w}_j - \eta \mathbf{b}_j. \tag{21}$$

Consequently, a standard implementation may access and update the entire classifier even though only a small subset of rows receives a nonzero gradient at the current iteration. To reduce computation,

Table 4: Comparison of the standard and sparse momentum updates.

	Standard SGD with momentum	Sparse momentum update
Active rows	$\mathbf{b}_j \leftarrow \beta \mathbf{b}_j + \mathbf{g}_j$ $\mathbf{w}_j \leftarrow \mathbf{w}_j - \eta \mathbf{b}_j$	$\mathbf{b}_j \leftarrow \beta \mathbf{b}_j + \mathbf{g}_j$ $\mathbf{w}_j \leftarrow \mathbf{w}_j - \eta \mathbf{b}_j$
Inactive momentum	$\mathbf{b}_j \leftarrow \beta \mathbf{b}_j$	Not updated
Inactive weight	$\mathbf{w}_j \leftarrow \mathbf{w}_j - \eta \mathbf{b}_j$	Not updated

Table 5: Per-iteration logit computation cost of different methods.

Method	Negatives per example	Logit computation cost
Full Softmax	$C-1$	$\mathcal{O}(BCd)$
BSGD (in-batch)	$< B$	$\mathcal{O}(B^2 d)$
SCENT	B	$\mathcal{O}(B^2 d)$
BSGD (KNN)	$0.1C$	$\mathcal{O}(BCd)$
PEACH	$L+B$ or $L+2B$	$\mathcal{O}(B(L+B)d)$

we instead use a row-sparse momentum update. For active rows $j \in \mathcal{A}_t$, we apply the standard momentum update,

$$\mathbf{b}_j \leftarrow \beta \mathbf{b}_j + \mathbf{g}_j, \qquad \mathbf{w}_j \leftarrow \mathbf{w}_j - \eta \mathbf{b}_j. \tag{22}$$

For inactive rows $j \notin \mathcal{A}_t$, both the classifier weight and its momentum state are left unchanged. The optimizer therefore touches only classifier rows involved in the current computation, reducing the number of operations from $\mathcal{O}(Cd)$ to $\mathcal{O}(|\mathcal{A}_t|d)$. Since $|\mathcal{A}_t| \ll C$ under sampled training, this sparse update is important for preserving the computational benefit of evaluating only a small subset of classes. Table 4 summarizes the difference from standard SGD with momentum.

C.4 Computational Cost

We set the uncached sampling budget to $m = B$ for all datasets, except for TreeOfLife-200M, where we use a larger budget of $m = 2B$, with $L \ll B \ll C$. PEACH evaluates L cached negatives and $B/2B$ sampled negatives for each example, resulting in $\mathcal{O}(B(L+B)d)$ logit computation cost per iteration. Full Softmax evaluates all C classes and therefore requires $\mathcal{O}(BCd)$ computation, whereas both BSGD (in-batch) and SCENT with B sampled negatives require $\mathcal{O}(B^2 d)$. For BSGD (KNN), the negative set is the union of the mined hard negative classes, in-batch labels, and randomly chosen labels fixed to a budget of $0.1C$. Table 5 summarizes the logit computation cost of the compared methods.

C.5 Memory Cost

PEACH maintains one auxiliary variable ν_i and L cached class identifiers for each training example. These per-example states are stored persistently in CPU memory rather than on GPU. With FP32 auxiliary variables and 32-bit class identifiers, the total CPU memory required over the full training set is

$$M_{\text{extra}} = n(4L+4) \text{ bytes}. \tag{23}$$

This cost does not depend on the classifier dimension d and the number of classes C, and does not increase the persistent GPU memory. At each iteration, only the auxiliary variables and cache IDs associated with samples in the current mini-batch are accessed and transferred to the GPU memory for parameter update. Table 6 reports the CPU memory ($L = 8$) required to store the auxiliary variable and cached class identifiers for different datasets.

D From lower variance to faster convergence

Proposition 2 characterizes negative-sampling variance for individual examples. In this section, we analyze how the combination of negative-sampling variance and mini-batch sampling variance affects

Table 6: CPU memory required to store the per-example auxiliary variable and cached IDs of PEACH ($L = 8$). Training examples denotes the number of examples used for training.

Dataset	Training examples	$\boldsymbol{\nu}$	Cache IDs	Total
TreeOfLife-10M	7,626,540	30.51 MB	244.05 MB	274.56 MB
Google Landmarks v2	3,306,331	13.23 MB	105.80 MB	119.03 MB
Glint360k	13,673,325	54.69 MB	437.55 MB	492.24 MB
TreeOfLife-200M	167,081,207	668.32 MB	5.35 GB	6.01 GB

convergence. Let

$$\mathbf{g}_t := \nabla_{\mathbf{W}} \Phi(\mathbf{W}_t, \boldsymbol{\nu}_t), \qquad \mathbf{G}_t^{\mathcal{B}} := \frac{1}{B} \sum_{i \in \mathcal{B}_t} \mathbf{G}_{i,t}, \qquad \widehat{\mathbf{G}}_t := \frac{1}{B} \sum_{i \in \mathcal{B}_t} \widehat{\mathbf{G}}_{i,t}. \tag{24}$$

Uniform mini-batch sampling without replacement and conditionally unbiased negative sampling imply

$$\mathbb{E}_{\mathcal{B}_t}[\mathbf{G}_t^{\mathcal{B}}] = \mathbf{g}_t, \qquad \mathbb{E}_{\mathcal{S}_t}[\widehat{\mathbf{G}}_t] = \mathbf{G}_t^{\mathcal{B}}. \tag{25}$$

Define mini-batch variance and negative sampling variance,

$$\sigma_{t,0}^2 := \mathbb{E}\big[\|\mathbf{G}_t^{\mathcal{B}} - \mathbf{g}_t\|_F^2\big], \qquad \sigma_{t,1}^2 := \mathbb{E}\Big[\|\widehat{\mathbf{G}}_t - \mathbf{G}_t^{\mathcal{B}}\|_F^2\Big], \tag{26}$$

and corresponding relative sampling variance for $\mathbf{g}_t \neq \mathbf{0}$:

$$\mathrm{P}_{t,0} := \frac{\sigma_{t,0}^2}{\|\mathbf{g}_t\|_F^2}, \qquad \mathrm{P}_{t,1} := \frac{\sigma_{t,1}^2}{\|\mathbf{g}_t\|_F^2} = \frac{\sum_{i=1}^n \rho_{i,t} \|\mathbf{G}_{i,t}\|_F^2}{Bn \|\mathbf{g}_t\|_F^2}. \tag{27}$$

Theorem 3 (Variance-dependent one-step descent)**.** *Assume that $\Phi(\cdot, \boldsymbol{\nu}_t)$ is L_t-smooth along the update $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \widehat{\mathbf{G}}_t$. If $\mathbf{g}_t \neq \mathbf{0}$, then*

$$\mathbb{E}[\Phi(\mathbf{W}_{t+1}, \boldsymbol{\nu}_t) \mid \mathcal{F}_t] \leq \Phi(\mathbf{W}_t, \boldsymbol{\nu}_t) - \eta \left[1 - \frac{L_t \eta}{2} \left(1 + \mathrm{P}_{t,0} + \mathrm{P}_{t,1}\right)\right] \|\mathbf{g}_t\|_F^2. \tag{28}$$

Consequently, a sufficient condition for positive expected one-step descent is

$$0 < \eta < \bar{\eta}_t := \frac{2}{L_t \left(1 + \mathrm{P}_{t,0} + \mathrm{P}_{t,1}\right)}. \tag{29}$$

At the same optimization state, PEACH and SCENT use the same data mini-batch scheme and therefore $\mathrm{P}_{t,0}$ is same; their difference is entirely captured by $\mathrm{P}_{t,1}$. Specifically,

$$\frac{\mathrm{P}_{t,1}^{\mathrm{PEACH}}}{\mathrm{P}_{t,1}^{\mathrm{unif}}} = \frac{\sum_{i=1}^n \rho_{i,t}^{\mathrm{PEACH}} \|\mathbf{G}_{i,t}\|_F^2}{\sum_{i=1}^n \rho_{i,t}^{\mathrm{unif}} \|\mathbf{G}_{i,t}\|_F^2}. \tag{30}$$

Hence a smaller relative per-sample variance $\rho_{i,t}^{\mathrm{PEACH}} < \rho_{i,t}^{\mathrm{unif}}$ indicates a smaller overall variance $\mathrm{P}_{t,1}^{\mathrm{PEACH}} < \mathrm{P}_{t,1}^{\mathrm{unif}}$. As a result, the tolerated learning rate $\bar{\eta}_t$ can be larger for PEACH than SCENT with uniform sampling, and hence a greater objective decrease per-iteration.

We define the maximum learning rate as the largest learning rate in our search grid for which the training loss remains finite and does not diverge throughout training. As shown in Table 7, PEACH supports a larger learning rate than SCENT (uniform), while remaining close to Full Softmax.

E Proofs for Section 5

E.1 Conditional Setup and Unbiasedness

We analyze the primal estimator conditional on the current optimization state and cache snapshot. Let $\mathcal{F}_t$ denote the sigma-algebra containing $(\mathbf{W}_t, \boldsymbol{\nu}_t)$ and the frozen caches $\{\mathcal{H}_{i,t}\}_{i=1}^n$. Throughout the analysis, $\boldsymbol{\nu}_t$ and the caches are treated as fixed before drawing the negative samples used for the

Table 7: Maximum learning rate across datasets.

Dataset	SCENT (uniform)	PEACH	Full Softmax
TreeOfLife-10M	2.0	5.0	5.0
Google Landmarks v2	0.02	0.5	1.0
Glint360k	2.0	5.0	10.0
TreeOfLife-200M	0.01	0.1	0.1

primal-gradient estimator. The delayed cache update in Algorithm 1 ensures that candidates observed at iteration t can affect the estimator only at a later visit.

For example i, define its exact full-class primal gradient as

$$\mathbf{G}_{i,t} = \frac{e^{-\nu_{i,t}}}{M} \sum_{j \neq y_i} e^{z_{ij}(\mathbf{W}_t)} \mathbf{\Delta}_{ij}. \tag{31}$$

Recall $|\mathcal{H}_{i,t}| = L$ and $R = M - L$. The PEACH estimator is

$$\widehat{\mathbf{G}}_{i,t}^{\mathrm{PEACH}} = \frac{e^{-\nu_{i,t}}}{M} \left[\sum_{j \in \mathcal{H}_{i,t}} e^{z_{ij}(\mathbf{W}_t)} \mathbf{\Delta}_{ij} + \frac{R}{m} \sum_{j \in \mathcal{S}_{i,t}^r} e^{z_{ij}(\mathbf{W}_t)} \mathbf{\Delta}_{ij} \right]. \tag{32}$$

Thus

$$\mathbb{E}\Big[\widehat{\mathbf{G}}_{i,t}^{\mathrm{PEACH}} \mid \mathcal{F}_t \Big] = \mathbf{G}_{i,t}. \tag{33}$$

E.2 A Finite-Population Sampling Identity

We first record the standard variance identity used below. Let $\mathbf{X}_1, \ldots, \mathbf{X}_N$ be a finite population of vectors or matrices with mean $\bar{\mathbf{X}} = N^{-1} \sum_{j=1}^N \mathbf{X}_j$, and let $\mathcal{S}$ be a uniformly sampled subset of size m without replacement. Then

$$\mathbb{E}\left[\left\| \frac{1}{m} \sum_{j \in \mathcal{S}} \mathbf{X}_j - \bar{\mathbf{X}} \right\|_F^2 \right] = \frac{N - m}{m(N-1)} \left(\frac{1}{N} \sum_{j=1}^N \|\mathbf{X}_j\|_F^2 - \|\bar{\mathbf{X}}\|_F^2 \right). \tag{34}$$

The factor $(N - m)/(N - 1)$ is the finite-population correction relative to independent sampling with replacement.

E.3 Proof of Proposition 1

Recall $\mathbf{\Delta}_{ij} = \nabla_{\mathbf{W}} z_{ij}(\mathbf{W}) = (\mathbf{e}_j - \mathbf{e}_{y_i})\mathbf{x}_i^\top$. Hence

$$\|\mathbf{\Delta}_{ij}\|_F^2 = 2\|\mathbf{x}_i\|_2^2, \qquad \langle \mathbf{\Delta}_{ij}, \mathbf{\Delta}_{ik} \rangle_F = \|\mathbf{x}_i\|_2^2 \quad \text{for } j \neq k. \tag{35}$$

Therefore,

$$\left\| \sum_{j \neq y_i} e^{z_{ij}} \mathbf{\Delta}_{ij} \right\|_F^2 = \|\mathbf{x}_i\|_2^2 A_i^2 (1 + \pi_i), \tag{36}$$

then we have

$$\|\mathbf{G}_i\|_F^2 = \frac{e^{-2\nu_i}}{M^2} \|\mathbf{x}_i\|_2^2 A_i^2 (1 + \pi_i). \tag{37}$$

For each negative class j, define

$$\mathbf{X}_{ij} := e^{-\nu_i} e^{z_{ij}} \mathbf{\Delta}_{ij}. \tag{38}$$

The population mean of $\{\mathbf{X}_{ij} : j \neq y_i\}$ is $\mathbf{G}_i$. Moreover,

$$\frac{1}{M} \sum_{j \neq y_i} \|\mathbf{X}_{ij}\|_F^2 = \frac{2e^{-2\nu_i}}{M} \|\mathbf{x}_i\|_2^2 A_i^2 \pi_i. \tag{39}$$

Applying Equation (34) with population size $N = C - 1 = M$ and sample size m gives

$$\mathbb{E}\Big[\|\widehat{\mathbf{G}}_i^{\text{unif}} - \mathbf{G}_i\|_F^2\Big] = \frac{M-m}{M-1}\frac{e^{-2\nu_i}\|\mathbf{x}_i\|_2^2 A_i^2}{mM^2}\left(2M\pi_i - 1 - \pi_i\right). \tag{40}$$

Dividing by $\|\mathbf{G}_i\|_F^2$ yields

$$\rho_i^{\text{unif}} = \frac{M-m}{M-1}\frac{2M\pi_i - 1 - \pi_i}{m(1+\pi_i)} = \frac{M-m}{M-1}\frac{2\kappa_i - 1 - \pi_i}{m(1+\pi_i)}. \tag{41}$$

When $m \ll C$, the finite-population correction is $\Theta(1)$. Since $\kappa_i = M\pi_i$, for large κ_i,

$$\rho_i^{\text{unif}} = \Theta\Big(\frac{\kappa_i}{m}\Big). \tag{42}$$

E.4 Proof of Proposition 2

Let

$$\mathcal{R}_i := [C] \setminus (\mathcal{H}_i \cup \{y_i\}), \qquad R := |\mathcal{R}_i| = M - L, \tag{43}$$

and define

$$A_i^{\text{unc}} = \sum_{j\in\mathcal{R}_i} e^{z_{ij}}, \qquad \pi_i^{\text{unc}} = \sum_{j\in\mathcal{R}_i}\left(\frac{e^{z_{ij}}}{A_i^{\text{unc}}}\right)^2, \qquad \kappa_i^{\text{unc}} = R\pi_i^{\text{unc}}. \tag{44}$$

Because the cached contribution is evaluated exactly, all sampling variance comes from the residual estimator:

$$\widehat{\mathbf{G}}_i^{\text{PEACH}} - \mathbf{G}_i = \frac{e^{-\nu_i}}{M}\left[\frac{R}{m}\sum_{j\in\mathcal{S}_i^r} e^{z_{ij}}\boldsymbol{\Delta}_{ij} - \sum_{j\notin\mathcal{H}_i,\, j\neq y_i} e^{z_{ij}}\boldsymbol{\Delta}_{ij}\right]. \tag{45}$$

Applying Equation (34) to the residual population of size R gives

$$\mathbb{E}\Big[\|\widehat{\mathbf{G}}_i^{\text{PEACH}} - \mathbf{G}_i\|_F^2\Big] = \frac{R-m}{R-1}\frac{e^{-2\nu_i}\|\mathbf{x}_i\|_2^2 (A_i^{\text{unc}})^2}{mM^2}\left(2R\pi_i^{\text{unc}} - 1 - \pi_i^{\text{unc}}\right). \tag{46}$$

Dividing by the exact gradient norm gives

$$\rho_i^{\text{PEACH}} = \frac{R-m}{R-1}\left(\frac{A_i^{\text{unc}}}{A_i}\right)^2\frac{2\kappa_i^{\text{unc}} - 1 - \pi_i^{\text{unc}}}{m(1+\pi_i)}. \tag{47}$$

E.5 Derivation of the Variance Ratio

Dividing the expression in Proposition 2 by that in Proposition 1 gives

$$\frac{\rho_i^{\text{PEACH}}}{\rho_i^{\text{unif}}} = \frac{(R-m)/(R-1)}{(M-m)/(M-1)}\left(\frac{A_i^{\text{unc}}}{A_i}\right)^2\frac{2R\pi_i^{\text{unc}} - 1 - \pi_i^{\text{unc}}}{2M\pi_i - 1 - \pi_i}, \tag{48}$$

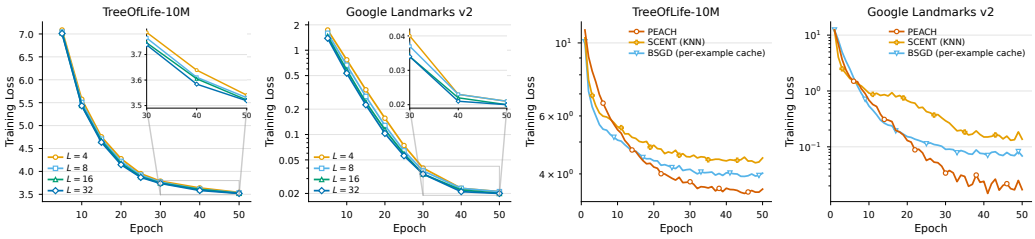
where $R = M - L$. When $L, m \ll C$, we have $R/M = \mathcal{O}(1)$ and the ratio of the two finite-population correction factors is $\mathcal{O}(1)$. Moreover, when the negative contributions are sufficiently concentrated such that $M\pi_i \gg 1$ and $R\pi_i^{\text{unc}} \gg 1$,

$$2M\pi_i - 1 - \pi_i = \Theta(M\pi_i), \qquad 2R\pi_i^{\text{unc}} - 1 - \pi_i^{\text{unc}} = \Theta(R\pi_i^{\text{unc}}). \tag{49}$$

Therefore,

$$\frac{\rho_i^{\text{PEACH}}}{\rho_i^{\text{unif}}} = \mathcal{O}\left(\left(\frac{A_i^{\text{unc}}}{A_i}\right)^2\frac{\pi_i^{\text{unc}}}{\pi_i}\right), \tag{50}$$

which implies the bound in Equation (19).



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

Figure 4: (a)(b): Effect of the per-example cache size L on training loss for TreeOfLife-10M and Google Landmarks v2. (c)(d): Training loss versus epoch for PEACH and two alternative uses of hard negatives on TreeOfLife-10M and Google Landmarks v2.

E.6 From Per-Example to Mini-Batch Variance

Recall

$$\mathbf{g}_t = \frac{1}{n}\sum_{i=1}^{n} \mathbf{G}_{i,t}, \qquad \mathbf{G}_t^{\mathcal{B}} = \frac{1}{B}\sum_{i\in\mathcal{B}_t} \mathbf{G}_{i,t}, \qquad \widehat{\mathbf{G}}_t = \frac{1}{B}\sum_{i\in\mathcal{B}_t} \widehat{\mathbf{G}}_{i,t}. \tag{51}$$

Suppose $\mathcal{B}_t$ is a uniformly sampled subset of B training examples without replacement. Applying Equation (34) to the finite population $\{\mathbf{G}_{i,t}\}_{i=1}^n$ gives

$$\sigma_{t,0}^2 := \mathbb{E}\big[\|\mathbf{G}_t^{\mathcal{B}} - \mathbf{g}_t\|_F^2 \mid \mathcal{F}_t\big] = \frac{n-B}{B(n-1)}\left(\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{G}_{i,t}\|_F^2 - \|\mathbf{g}_t\|_F^2\right). \tag{52}$$

Equivalently,

$$\sigma_{t,0}^2 = \frac{n-B}{B(n-1)}\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{G}_{i,t} - \mathbf{g}_t\|_F^2. \tag{53}$$

We next characterize the variance from negative sampling. Since each per-example estimator is conditionally unbiased, taking expectation gives

$$\sigma_{t,1}^2 := \mathbb{E}\Big[\|\widehat{\mathbf{G}}_t - \mathbf{G}_t^{\mathcal{B}}\|_F^2 \mid \mathcal{F}_t\Big] = \frac{1}{Bn}\sum_{i=1}^{n}\rho_{i,t}\|\mathbf{G}_{i,t}\|_F^2. \tag{54}$$

Therefore,

$$\mathrm{P}_{t,1} := \frac{\sigma_{t,1}^2}{\|\mathbf{g}_t\|_F^2} = \frac{\sum_{i=1}^{n}\rho_{i,t}\|\mathbf{G}_{i,t}\|_F^2}{Bn\|\mathbf{g}_t\|_F^2}. \tag{55}$$

Finally, noticing

$$\widehat{\mathbf{G}}_t - \mathbf{g}_t = (\widehat{\mathbf{G}}_t - \mathbf{G}_t^{\mathcal{B}}) + (\mathbf{G}_t^{\mathcal{B}} - \mathbf{g}_t), \tag{56}$$

and $\mathbb{E}[\widehat{\mathbf{G}}_t - \mathbf{G}_t^{\mathcal{B}} \mid \mathcal{B}_t, \mathcal{F}_t] = \mathbf{0}$. Hence,

$$\mathbb{E}\Big[\|\widehat{\mathbf{G}}_t - \mathbf{g}_t\|_F^2 \mid \mathcal{F}_t\Big] = \sigma_{t,0}^2 + \sigma_{t,1}^2. \tag{57}$$

F Ablation Study

F.1 Cache Size

We vary the per-example cache size over $L \in \{4, 8, 16, 32\}$ on TreeOfLife-10M and Google Landmarks v2 while keeping all other hyperparameters fixed. Fig. 4a and 4b show that PEACH is insensitive to the cache size: a small cache achieves convergence close to that of substantially larger caches, with only modest gains from increasing L. This is consistent with the concentration observed in Fig. 1b, where only a few hard negatives account for most of the negative partition. We use $L = 8$ by default as a balance between optimization performance and cache overhead.

Table 8: Hyperparameters of different methods on different datasets. Entries with "-" indicate that the corresponding hyperparameter is not used.

Dataset	Hyper.	Full Softmax	BSGD (in-batch)	SCENT (uniform)	SCENT (in-batch)	BSGD (KNN)	PEACH
TreeOfLife-10M	lr	5.0	2.0	2.0	2.0	5.0	5.0
	α	-	-	e^{10}	e^{10}	-	e^{11}
Google Landmarks v2	lr	1.0	0.5	0.02	0.2	0.5	0.5
	α	-	-	e^{13}	e^{13}	-	e^{13}
Glint360k	lr	10.0	2.0	2.0	2.0	5.0	5.0
	α	-	-	e^{10}	e^{9}	-	e^{12}
TreeOfLife-200M	lr	0.1	0.1	0.01	0.1	0.1	0.1
	α	-	-	e^{11}	e^{11}	-	e^{13}

F.2 Baselines with different sampling strategies

To examine whether hard-negative selection alone accounts for PEACH's improvement, we compare two alternative ways of using hard negatives. BSGD (per-example cache) incorporates a per-example cache into BSGD; its gradient remains biased because the estimated normalizer appears inside the logarithm. SCENT (KNN) replaces uniform negative sampling with KNN-selected classes without correcting for their nonuniform selection, so it also loses the unbiased-gradient guarantee. As shown in Fig. 4c and 4d, PEACH achieves the lowest training loss on both TreeOfLife-10M and Google Landmarks v2. Thus, neither adding a cache to BSGD nor adding KNN selection to SCENT matches the convergence observed when PEACH combines an exact cached contribution with unbiased estimation of the remaining negatives.

G Training Details

G.1 Hyperparameters

For all methods, we set batch size $B = 4096$, and use a cosine decay learning rate schedule for 50 epochs. Unless otherwise specified, we use SGD with momentum 0.9. The hyperparameters are presented in Table 8.

SCENT We follow a similar setting in Wei et al. (2026b). We tune the learning rate α of the dual variable by searching the value of $\log(\alpha)$ from 5 to 15.

PEACH In addition to SCENT, we set cache size $L = 8$, and uncached budget m to 4096 for TreeOfLife-10M, Google Landmarks v2 and Glint360k while 8192 for TreeOfLife-200M. Each per-example cache $\mathcal{H}_i$ is randomly initialized from its negative label space. We use a learning-rate warmup for the first 5 epochs in training to stabilize initial cache construction.

KNN We follow a similar setting in Song et al. (2020) and build a class-level KNN graph over the classifier weights with $K = 12$ neighbors per class, computed exactly and rebuilt at the start of every epoch. At each iteration, the active set is the union of the batch labels and their graph neighbors, fixed to a budget of $m_{\text{KNN}} = 0.1C$ classes. The first 5 epochs use the batch labels randomly padded to the same budget m_{KNN}.

G.2 Hardware And Runtime

We run our experiments on 4 NVIDIA A100 GPUs for all methods, each with 80 GB of GPU memory. In principle, our experiments can also be performed on a single GPU setup, but it would take longer to train. Table 9 reports the training wall-clock times for each dataset.

Table 9: Running time (hours) for 50 epochs across four datasets.

Method	TreeOfLife-10M	Google Landmarks v2	Glint360k	TreeOfLife-200M
Full Softmax	0.32	2.08	3.53	65.69
BSGD (KNN)	0.18	0.58	1.39	12.80
BSGD (in-batch)	0.16	0.32	1.07	8.00
SCENT (in-batch)	0.17	0.33	1.10	8.05
SCENT (uniform)	0.17	0.34	1.12	8.08
PEACH	0.19	0.42	1.39	9.80

H Other Experiments Details

H.1 Empirical Observations in Section 3

We randomly sample 1000 training examples from randomly selected 100 classes on Google Landmarks v2. The same probe set is used for all checkpoints trained by Full Softmax method.

H.2 Relative sampling variance in Section 6.2

We evaluate the relative sampling variance ρ_i defined in Equation (14) at several frozen checkpoints trained on PEACH. For each dataset, we use a fixed set of 10000 training examples throughout the evaluation and report the average ρ_i across these examples.

For fair comparison, we use $m = 4096$ as the negative-class budget for all estimators. Uniform sampling draws a shared set of m classes without replacement from full label space, excluding the positive label for each example. For PEACH, cached negatives are included exactly, and the remaining budget is used to sample from the uncached classes without replacement.

H.3 Cache Quality Evaluation in Section 6.2

We evaluate cache quality using frozen checkpoints and a fixed set of 10000 training examples for each dataset. For every probe example, we compute the full logits over all classes and measure the uncached partition ratio A_i^{unc}/A_i using the cache saved at that checkpoint. We also compute the same ratio for the exact top-L negatives under the current classifier as a reference.