ANN Softmax: Acceleration of Extreme Classification Training

Kang Zhao, Liuyihan Song, Yingya Zhang, Pan Pan, Yinghui Xu, Rong Jin
Machine Intelligence Technology Lab, Alibaba Group
Beijing, China
{zhaokang.zk, liuyihan.slyh, yingya.zyy, panpan.pp, renji.xyh, jinrong.jr}@alibaba-inc.com

ABSTRACT

Thanks to the popularity of GPU and the growth of its computational power, more and more deep learning tasks, such as face recognition, image retrieval and word embedding, can take advantage of extreme classification to improve accuracy. However, it remains a big challenge to train a deep model with millions of classes efficiently due to the huge memory and computation consumption in the last layer. By sampling a small set of classes to avoid the total classes calculation, sampling-based approaches have been proved to be an effective solution. But most of them suffer from the following two issues: i) the important classes are ignored or only partly sampled, such as the methods using random sampling scheme or retrieval techniques of low recall (e.g., locality-sensitive hashing), resulting in the degradation of accuracy; ii) inefficient implementation owing to incompatibility with GPU, like selective softmax. It uses hashing forest to help select classes, but the search process is implemented in CPU. To address the above issues, we propose a new sampling-based softmax called *ANN Softmax* in this paper. Specifically, we employ binary quantization with inverted file system to improve the recall of important classes. With the help of dedicated kernel design, it can be totally parallelized in mainstream training framework. Then, we find the size of important classes that are recalled by each training sample has a great impact on the final accuracy, so we introduce sample grouping optimization to well approximate the full classes training. Experimental evaluations on two tasks (Embedding Learning and Classification) and ten datasets (e.g., MegaFace, ImageNet, SKU datasets) demonstrate our proposed method maintains the same precision as Full Softmax for different loss objectives, including cross entropy loss, ArcFace, CosFace and D-Softmax loss, with only 1/10 sampled classes, which outperforms the state-of-the-art techniques. Moreover, we implement *ANN Softmax* in a complete GPU pipeline that can accelerate the training more than 4.3×. Equipped our method with a 256 GPUs cluster, the time of training a classifier of 300 million classes on our SKU-300M dataset can be reduced to ten days.

PVLDB Reference Format:
Kang Zhao, Liuyihan Song, Yingya Zhang, Pan Pan, Yinghui Xu, Rong Jin. ANN Softmax: Acceleration of Extreme Classification Training. PVLDB, 15(1): 1-10, 2022.
doi:10.14778/3485450.3485451

Proceedings of the VLDB Endowment, Vol. 15, No. 1 ISSN 2150-8097.
doi:10.14778/3485450.3485451

1 INTRODUCTION

In the past decades, deep learning has achieved huge success in many areas, such as computer vision [25, 38, 41, 46, 47], speech recognition [29] and natural language processing (NLP) [9, 10, 27]. Among them, extreme classification is playing an increasingly important role, because of the popularity of GPU and the growth of its computational power. Many literatures [36, 39, 43] have demonstrated that a lot of tasks can be modeled as an extreme classification problem to increase the precision. Take the embedding learning in face recognition as an example. In order to make the features of each person more discriminative, we can add a large number of face identities into the dataset to increase the information that model can learn [5, 37]. Simultaneously, it makes the scale of face classification larger and larger. Another example is product image classification. At Alibaba, we build a huge fine-grained (i.e., stock keeping unit (SKU) level) retail product image dataset, which contains several billions of samples in hundreds of millions of classes. To improve the recognition ability of product images, we expand the number of product classification from one million to tens of millions, or even hundreds of millions.

Although extreme classification can be applied to boost model accuracy, there still remains a big challenge due to the huge memory and computation consumption in the last layer. Given the hybrid parallel training framework (an efficient framework to solve extreme classification in community) [2, 8, 35], we conduct experiments by using 128 Tesla V100-32G GPUs to train a classifier of 100 million classes, the batch size of one GPU is 48 and the dimension of weight matrix in last layer is 512, then we have:

$$Mem_{weight} = \frac{100000000}{128} \times 512 \times 4\ Bytes = 1.49\ GB, \quad (1)$$

$$Mem_{output} = 128 \times 48 \times \frac{100000000}{128} \times 4\ Bytes = 17.8\ GB. \quad (2)$$

It's very clear the output matrix of the last layer occupy the majority of memory. And we find in each iteration, almost 80% of the time is spent on the forward/backward stage of the last layer.

To tackle this problem, many sampling-based methods have been proposed to reduce the output memory and computation of the last layer, by sampling a small set of classes from the total classes. [2] adopts random sampling scheme, which can not guarantee the important classes are sampled. An alternative way is using approximate nearest neighbor search (ANN) to select classes, such as [26, 45]. But the recall of their retrieval algorithms are not high enough, resulting in only a small part of important classes being sampled. Consequently, these two kinds of methods have more or less accuracy degradation. What's worse, most ANN-based sampled methods are dismissed in practice owing to their incompatibility with GPU.

In this paper, we propose a new sampled softmax called *ANN Softmax* to address above issues. Different from methods that sacrifice accuracy for high sampling rate, we pursue the same precision with Full Softmax and the actual speedup benefit in practice. The main contributions of our work are outlined as follows:

- We propose a novel sampling approach: binary quantization with inverted file system (IVF-BQ). With the help of dedicated kernel design, it can be totally parallelized in mainstream training framework, like PyTorch [28].
- Based on an important observation that the important class recalled by each training sample has a great influence on the final training accuracy, we employ two kinds of sample grouping optimization to well approximate the Full Softmax.
- As an ANN-based way, we implement our method in a complete GPU pipeline, which can accelerate the training more than 4.3×.
- Experimental evaluations on broad range of tasks and datasets show our proposed method maintains the same accuracy as full classes training for different loss objectives with only 1/10 sampled classes, outperforming the state-of-the-art techniques.

2 PRELIMINARIES

To facilitate our discussion, we divide a typical extreme classification task with softmax loss into three parts, and give some notations below:

I. Backbone network for feature extraction:

$$\mathbf{x} = f(I; \theta), \tag{3}$$

where I is the input image, θ is the parameter of the backbone, and $\mathbf{x} \in \mathrm{R}^d$ denotes the extracted feature.

II. A fully connected (FC) layer:

$$\mathbf{o} = h(\mathbf{x}; \mathbf{W}, \mathbf{b}). \tag{4}$$

Let N be the total number of classification, $\mathbf{W} \in [N, d]$[1] is denoted as the weight parameters of FC layer, where each row $\mathbf{w}_j \in \mathrm{R}^d$ represents the weight vector of the j-th class, and $\mathbf{b}$ is bias. For simplicity, we set $\mathbf{b} = 0$ as in [2, 8]. Then we have $\mathbf{o} = \mathbf{W}\mathbf{x} \in \mathrm{R}^N$.

III. Softmax Cross-Entropy Loss:

$$l(\mathbf{x}) = -\log \frac{\exp(\mathbf{o}_y)}{\sum \exp(\mathbf{o})}, \tag{5}$$

where y is the label of $\mathbf{x}$, and $\mathbf{o}_y = \mathbf{x}^\top \mathbf{w}_y$.

Denote the batch size (of one GPU) as B, and $\mathbf{X} \in [B, d]$ as the feature of mini-batch. The matrix multiplication (mm) in FC layer ($\mathbf{O} = \mathbf{X}\mathbf{W}^\top \in [B, N]$) has the complexity $\mathcal{O}(BN)$, and the softmax loss also has multiple operations with complexity $\mathcal{O}(BN)$ on $\mathbf{O}$ matrix, such as the *exp*, *sum* and *div* calculations. It is obvious that when N is quite large, not only the GPU memory for $\mathbf{O}$ is very huge, but also the computations (including forward and backward) of FC layer and softmax loss are very time consuming, as we mentioned above.

[1] $[N, d]$ is short for $\mathrm{R}^{N \times d}$.

Therefore, decreasing the cost caused by FC layer and softmax loss is a key issue in dealing with the challenges in extreme classification. Existing methods can be roughly divided into two major categories: approximate softmax methods and sampling-based methods.

2.1 Approximate softmax methods

These methods focus on the approximation of softmax function. As a pioneer, Hierarchical Softmax (HSM) [16] turns a multi-class classifier to a hierarchical binary classifier, by building a tree with prior class distribution. Although hierarchical structure can reduce the computation, it has a negative impact on the accuracy. [18] proposes Noise Contrastive Estimation (NCE) based on a similar idea: transforming the multi classification problem into a binary logistic regression, which also brings about inferior performance. A simple hashing based divide-and-conquer algorithm is presented in Merged-Average Classification via Hashing (MACH) [31] to solve the multi-class classification problem. However, this method still can not achieve the same performance as Full Softmax. The key idea of [44] is training a slimmed model with a Random Projected (RP) softmax classier first, and then recovering it to the original version. There exists an error between the recovered classier and the original classier.

2.2 Sampling-based methods

As the term suggests, one can sample a small set of classes from the total classes to avoid full class calculations. There are two common sampling ways: frequency-based sampling and classes-based sampling.

2.2.1 *Frequency-based sampling.*

In the area of NLP, most of words (classes) are low frequency, high frequency words (classes) only account for a small part in datasets, which means we can just use the high-frequency words for classification tasks, turning extreme classification to a normal scale classification [6, 34]. But in practical applications, such as face classification, all classes should be treated equally. We can not do the sampling based on frequency.

2.2.2 *Classes-based sampling.*

Random sampling is a typical classes-based sampling method. It is combined with hybrid parallel framework in Partial FC [2] to extend classification to 100 million level. Softmax Dissection (D-Softmax) [20] proposes random sampling variants (D-Softmax-B/K) to accelerate its training process. Nevertheless, all methods above have different degrees of precision loss, because random sampling can not guarantee important classes are sampled.

Some literatures [26, 40, 45] have shown that the classes having larger responses with $\mathbf{x}$ are more important than that having smaller ones, because classes with larger responses will have larger gradients in the backward propagation. So sampling these important classes is an effective way to approximate Full Softmax. Selective softmax [45] builds a hashing forest (also called HF-Softmax) to partition the $\mathbf{W}$ into small cells, then searches the important classes in several cells. A fast locality-sensitive hashing technique (LSH-Softmax) in [26] is adopted to approximate the actual dot product. Whatever it's hashing forest [30] or locality-sensitive hashing

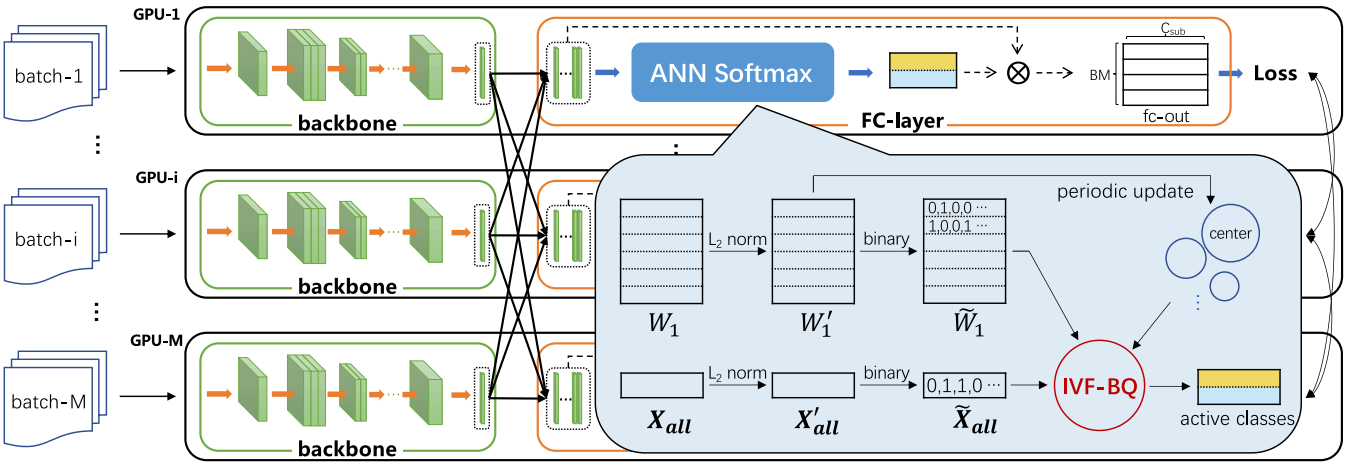

Figure 1: The hybrid parallel training framework of ANN Softmax.

(LSH) [14, 15], their search recall is relatively low during the ANN algorithms [3, 4, 13, 48, 49]. And both methods are not totally implemented by GPU. Our method can be categorized into this scope, and we refer to important classes as active classes in the rest of this paper, keeping consistent with [45].

k-Nearest Neighbor Softmax (KNN Softmax) [35] is a variant, which constructs a global graph on $\mathbf{W}$ and uses y (instead of $\mathbf{x}$) to select active classes. However, it has an apparent limitation: the graph storage and the time spent on brute-force graph building will increase by $\mathcal{O}(N^2)$, preventing it from extending to a larger scale.

3 ANN SOFTMAX

This section describes the formulation of our ANN Softmax. First, we present the training architecture based on distributed GPUs. Then we introduce IVF-BQ for active classes selection and its kernel design, followed by two kinds of sample grouping optimization. Finally, a complete pipeline of ANN Softmax is given.

3.1 Training Architecture

Similar to [8, 35], we employ the hybrid parallel training framework as shown in Figure 1. Suppose the number of GPUs is M, each GPU has a complete backbone network (called data-parallel), and $1/M$ weight parameters of FC layer (called model-parallel). We denote the FC weight in i-th GPU as $\mathbf{W}_i \in [C, d]$, where $C = N/M$.

In the forward pass, GPU-i uses the backbone network to extract the features of batch-i, then All-Gather features (denoted as $\mathbf{X}_{all} \in [BM, d]$) from GPU-1 to GPU-$M$. After that, we use ANN Softmax module to calculate the active classes, which size is C_{sub} ($C_{sub} < C$), and perform the inner product between $\mathbf{X}_{all}$ and the active classes to get the output (denoted as $fc\text{-}out$), followed by the loss computation. We call $r = C_{sub}/C$ the sampling rate.

In the backward pass, we will get the gradient of $fc\text{-}out$ first. According to the chain rule, we can derive the gradient of active classes, then expand it to the gradient of $\mathbf{W}_i$, by setting the gradient of unselected classes to zero. Similarly, we can compute the gradient of $\mathbf{X}_{all}$. Considering $\mathbf{X}_{all}$ only do inner product with $\mathbf{W}_i$, instead of total $\mathbf{W}$, we need to add an All-Reduce Communication to get the complete gradient of $\mathbf{X}_{all}$. Finally, each GPU takes its corresponding gradient in the order of All-Gather and executes the backward propagation of the backbone.

ANN Softmax module mainly includes active classes selection and sample grouping optimization, we will introduce them in order.

3.2 Active Classes Selection

For the sake of clarity, let's consider the training of a single sample $\mathbf{x}$ first. As stated in [45], the active classes are the k classes that have largest inner product responses with $\mathbf{x}$. So, it can be regarded as an ANN search problem of using $\mathbf{x}$ to search for k nearest classes among the C weight vectors (how to choose k will be described in section 3.3). The more top k classes are recalled, the more large gradients are preserved [45], then the smaller the accuracy difference from Full Softmax will be. We summarize the challenges we face in ANN-based active classes selection as follows: 1) high recall of ANN algorithm; 2) friendly implementation of GPU; 3) efficient search process (since each training iteration will execute one ANN search). Therefore, we propose the IVF-BQ strategy.

3.2.1 *IVF-BQ.*

Following [2, 35], we perform L_2 normalization on $\mathbf{x}$ and $\mathbf{W}_i$, making the Euclidean distance and inner product equivalent. Inspired by IVF-PQ [22], we execute K-means clustering on $\mathbf{W}'_i$ (the L_2 normalization of $\mathbf{W}_i$), and the number of centers is $center_{num}$. The cluster centers are the entries of the inverted lists (one inverted list contains all data that belong to the center). One can visit very few inverted lists to greatly reduce the scope of the search. Noted that: i) The centers should also be L_2 normed to ensure they are in the same metric space as $\mathbf{x}'$. ii) Clustering is only performed within local weight parameters, i.e., no communications between GPUs are needed, and K-means clustering can be easily implemented by GPU. iii) Considering the weight parameters change gradually, we will update the centers periodically (the interval is denoted as T) to make the clustering time be ignored during the training process.

Besides the inverted lists, we perform mean binarization of $\mathbf{x}'$ and $\mathbf{W}'_i$ to further accelerate the ANN search, by replacing Euclidean distance with Hamming (HM) distance. Specifically, we take three

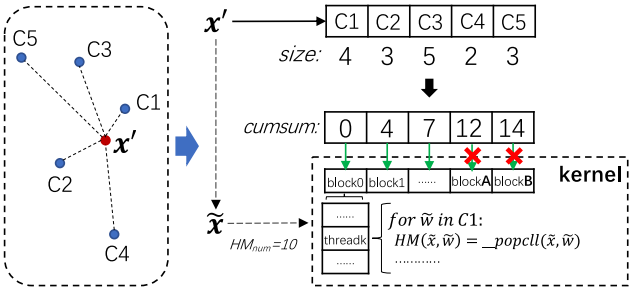

Figure 2: The kernel design of IVF-BQ.

steps:

Step 1: Calculate the mean vector ($\in \mathrm{R}^d$) of $\mathbf{W}'_i$;

Step 2: Based on the mean vector, we binarize $\mathbf{W}'_i$ and $\mathbf{x}'$ as follows: if one value in $\mathbf{W}'_i$ / $\mathbf{x}'$ is greater than the mean, then its binary expression is 1, otherwise is 0;

Step 3: Convert the binary representations of $\mathbf{W}'_i$ and $\mathbf{x}'$ (produced by Step 2) into byte streams to save storage, which only takes 1/32 memory compared with data in float.

We denote the binary result of $\mathbf{x}'$ as $\widetilde{\mathbf{x}}$, and one binary weight vector as $\widetilde{\mathbf{w}}$.

In the search process, we first calculate the distance between $\mathbf{x}'$ and cluster centers, and select the inverted lists closest to $\mathbf{x}'$. Then we calculate HM distance between $\widetilde{\mathbf{x}}$ and all $\widetilde{\mathbf{w}}$ in the selected inverted lists to sort out the largest k' ($k' > k$) results. Although binary representation can speed up the calculation, its information loss will lead to the decline of ANN recall. Therefore, we will rerank k' results based on original features to get the final top k results, which takes both efficiency and recall into account.

Moreover, considering the size of each inverted list is different, we fix the number of $\widetilde{\mathbf{w}}$ (denoted as HM_{num}), not the number of cluster centers [11, 12, 22], that will be visited by $\widetilde{\mathbf{x}}$ in the implementation. This scheme can keep the search range of each training sample almost the same, and is helpful for the stability of training.

3.2.2 *Kernel Design.*

Because $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{w}}$ are stored in byte stream (we need to call *__popcll* instruction to compute the Hamming distance), and the number of inverted list that will be visited is not fixed (see section 3.2.1), it is very difficult to implement IVF-BQ with existing interfaces in the mainstream deep learning frameworks (such as PyTorch, TensorFlow [1]). To tackle this problem, we add a new CUDA kernel function into PyTorch framework.

For simplicity, suppose there are five centers and let HM_{num} = 10, as shown in Figure 2. The nearest center of $\mathbf{x}'$ is C1, followed by C2, C3, C4 and C5, and their sizes are in the array {4, 3, 5, 2, 3}. We will first calculate the prefix sum of this array, and store the result into an array called *cumsum*: {0, 4, 7, 12, 14}. The first value of *cumsum* is always 0, means there is no data in front of C1. The second value is 4, means the center in front of C2 (i.e., C1) has 4 data points. The third value is 7, means the centers in front of C3 (i.e., C1 and C2) have 7 data points, and so on. So the size of *cumsum* is the same as the number of centers.

At the kernel design, we will assign a block to each inverted list that $\widetilde{\mathbf{x}}$ will visit. One block will read the corresponding value in the *cumsum* array, and compare it with HM_{num}. If the value is less than HM_{num}, we will traverse the inverted list, and use all threads in the block to compute the Hamming distances between $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{w}}$ in parallel, because less than HM_{num} data points in front of this inverted list have been visited, e.g., block0 and block1 in Figure 2. Otherwise (the value is larger than or equal to HM_{num}), the block will exit without visiting the inverted list. For example, Figure 2 shows that blockA and blockB will read number 12 and 14, respectively. 12 and 14 are both bigger than 10 (HM_{num}), so the two blocks will not be executed.

In addition, considering the principle of locality, we will rearrange the memory of $\widetilde{\mathbf{w}}$ to make the $\widetilde{\mathbf{w}}$ in one inverted list are memory continuous. When one block traverse all $\widetilde{\mathbf{w}}$ in one inverted list, the memory continuity of $\widetilde{\mathbf{w}}$ will speed up the Hamming distances calculation more than 2×.

Although we also use ANN to search active classes, our approach is totally different from HF-Softmax and LSH-Softmax in two aspects: 1) We propose a new active classes selection method called IVF-BQ, which is completely GPU implemented. HF/LSH-Softmax all require the CPU to execute the neighborhood retrieval; 2) Our IVF-BQ has higher recall than hashing forest and LSH algorithms, which is important for the final training accuracy. We prove this in ablation study. By the way, we adopt IVF-BQ instead of IVF-PQ since Hamming distance calculation in BQ is generally efficient than table lookups in PQ.

Algorithm 1 Naive Sample Grouping Optimization

Input: The total batch $\mathbf{X}'_{all} \in [BM, d]$; the number of groups g_{num}; the size of active classes C_{sub}; weight parameters $\mathbf{W}'_i$ in one GPU;

Output: The output of FC layer $fc\text{-}out \in [BM, C_{sub}]$

$\{Xgroup_j\}_{j=0}^{g_{num}-1}$ = split($\mathbf{X}'_{all}$, g_{num}); // $Xgroup_j \in [\frac{BM}{g_{num}}, d]$
$fc\text{-}out$ = [];
for $j \leftarrow 0$ to $g_{num} - 1$ **do**
  $l(Xgroup_j)$ = IVF-BQ($Xgroup_j$);
  $l(Xgroup_j)$ = duplicate($l(Xgroup_j)$);
  $l_{random} = \emptyset$;
  if size($l(Xgroup_j)$) < C_{sub} **then**
    l_{random} = sample($\mathbf{W}'_i$, C_{sub}− size($l(Xgroup_j)$));
  end if
  $l(Xgroup_j) = l(Xgroup_j) + l_{random}$;
  $\mathbf{W}_{active}$ = get C_{sub} weights from $\mathbf{W}'_i$ based on $l(Xgroup_j)$;
  $fc\text{-}out$.append(mm($Xgroup_j$,$\mathbf{W}^{\top}_{active}$));
end for
$fc\text{-}out$ = concat($fc\text{-}out$);
return $fc\text{-}out$;

3.3 Sample Grouping Optimization

In large scale training, we will apply ANN Softmax to the total batch samples $\mathbf{X}_{all}$. Let $l(x)$ be the active classes indices of one sample $\mathbf{x}$, then $l(X_{all})$ is a union of $l(x)$. We can get the active classes $\mathbf{W}_{active}$ according to $l(X_{all})$, and do the mm operation between $\mathbf{W}_{active}$ and $\mathbf{X}_{all}$. As mentioned above, $\mathbf{X}_{all}$ in the hybrid parallel framework contain $B \times M$ samples. If we use 128 GPUs (M = 128) to train a 100 million classifier with B = 32, k is set to 300, the overlap

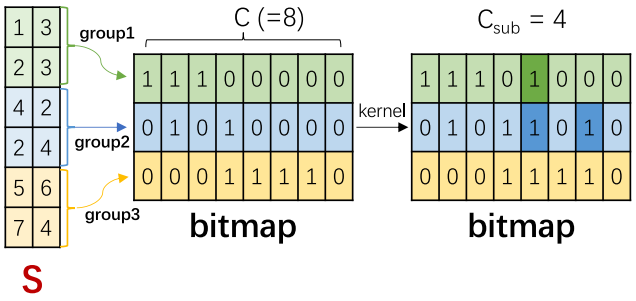

Figure 3: The bitmap flow in parallel implementation. We assume $C = 8$, $C_{sub} = 4$, $g_{num} = 3$ and $\mathbf{S} \in [6, 2]$. S will be divided into three groups: {row1, row2}, {row3, row4} and {row5, row6}. After duplication, {row1, row2} group has three indices in total, so the first row of bitmap has three 1. We randomly add one 1 in the first row to make its sum equal to C_{sub}.

rate of union is assumed to be 50%[2], then we find the size of $l(X_{all})$ is almost equal to the number of classification in one GPU:

$$32 \times 128 \times 300 \times 0.5 \approx 10^8/128. \tag{6}$$

That is to say, the size of $\mathbf{W}_{active}$ ($= C_{sub}$) is close to $\mathbf{W}_i$ ($= C$), which goes against our intension: we want to make C_{sub} small enough to improve computation efficiency and save the storage of $fc\text{-}out$.

In general, one can reduce the size of $l(x)$, i.e., the value of k, to make the size of $l(X_{all})$ smaller. Unfortunately, we empirically find that the value of k has a great influence on the final training accuracy (see the ablation study in section 4.1.2). For a training sample $\mathbf{x}$, besides positive class ($\mathbf{w}_y$), the majority of its active classes can be seen as hard negative classes[3]. The larger N or C is, the more the number of hard negative classes will be. If we reduce the active classes of training samples, the corresponding hard negative classes will be reduced, and the difference between ANN and Full Softmax will become larger, resulting in the decline of the final accuracy. What's worse, if we formulate k as follows (duplication is ignored for clarity):

$$k = \frac{C_{sub}}{B \times M}, \tag{7}$$

further reduction in C_{sub} (e.g. from $\frac{1}{10}C$ to $\frac{1}{100}C$), or increasing the number of GPUs ($B \times M$ will be increased), may cause k less than 1. It means some training samples have no hard negative classes at all, leading to the decrease of their classification accuracy.

3.3.1 *A Naive Solution.*

To solve this problem, a naive solution would be: divide the total batch ($\mathbf{X}_{all}$) into multiple groups, so that the number of each group active classes is small enough. We call one group as group-batch and the number of groups is denoted as g_{num}, then we have the following formula:

$$k = \frac{C_{sub}}{B \times M} \times g_{num}. \tag{8}$$

It can be seen k becomes g_{num} times larger than it in Eq (7). We execute the active classes selection on each group-batch, and concat their results to get $fc\text{-}out$ at last. If the group-batch has less than C_{sub} active classes (due to duplication), we will randomly sample the left weight from $\mathbf{W}'_i$. Algorithm 1 summarizes the naive solution of sample grouping optimization.

Algorithm 2 ANN Softmax

Input: The total batch $\mathbf{X}'_{all} \in [BM, d]$; the label of total batch Y_{all}; the number of groups g_{num}; the size of active classes C_{sub}; weight parameters $\mathbf{W}'_i$ in one GPU; the bitmap $BP \in [g_{num}, C]$
Output: The output of FC layer $fc\text{-}out \in [BM, C_{sub}]$

1: Initialize $BP = 0$;
2: $l_{positive}$ = get positive class indices based on Y_{all};
3: S = IVF-BQ($\mathbf{X}'_{all}$);
4:
5: // fill S into BP according to group relationship
6: BP = fill(BP, S);
7: // fill $l_{positive}$ into BP according to group relationship
8: BP = fill($BP, l_{positive}$);
9: // make the sum of each row of BP is C_{sub}
10: BP = sample_kernel(BP, C_{sub});
11:
12: $tempW = \mathbf{W}'_i[mask\_select(BP), :]$;
13: $tempX = \mathbf{X}'_{all}.reshape(g_{num}, \frac{BM}{g_{num}}, d)$;
14: $fc\text{-}out = bmm(tempX, tempW.transpose(2, 1))$;
15:
16: **return** $fc\text{-}out$;

3.3.2 *Parallel Implementation.*

Although the naive solution is easy to be implemented, there exists one problem: the ANN Softmax process of each group-batch will be run one by one. For the hardware like GPU, serial implementation can not make full use of computing resources. As a result, we propose a parallel optimization method to accelerate the process.

First, IVF-BQ is performed on $\mathbf{X}'_{all}$ in parallel, we mark the output matrix as $\mathbf{S} \in [BM, k]$. Then we create a 2D bitmap in the shape of $[g_{num}, C]$, and fill S into this bitmap according to group mapping relationship. As shown in Figure 3, the row 1 and 2 in S are the first group, so they are filled into the line 1 of bitmap. Bitmap[0][0] = 1 means in the first group, $\mathbf{w}_0$ is selected as active class. To make different group-batches have the same size of active classes, the sum of each row of bitmap should be equal to C_{sub}. If some group-batches have less than C_{sub} active classes, we will randomly add active classes indices to their rows, as we did in naive solution. Differently, we will do parallel random sampling on this bitmap, which can be easily implemented by a CUDA kernel.

After that, we will use the bitmap to select out a 3D weight from $\mathbf{W}'_i$: line 12 in Algorithm 2, where $mask\_select$ converts the bits to indices, and $tempW$ is in the shape of $[g_{num}, C_{sub}, d]$. A similar shape transformation is applied to $\mathbf{X}'_{all}$: line 13 in Algorithm 2, where $tempX$ is in the shape of $[g_{num}, \frac{BM}{g_{num}}, d]$. Finally, we execute

[2]The overlap rate will change in pace with the training, from a large value to a small value. We take 0.5 as its mean value here.
[3]"hard" means the negative classes in active classes have larger inner product response, compared with that are not in active classes.

batch matrix multiplication (bmm) operation on $tempW$ and $tempX$ to get the final output (instead of mm). If not specified, the parallel grouping optimization is used in the following experiments.

We summarize the total pipeline of ANN Softmax in Algorithm 2. Line 2~3 show that we use the label of $\mathbf{X}'_{all}$ ($\mathbf{Y}_{all}$) to select positive classes, and IVF-BQ function to choose active classes. It is noted that active classes may overlap with positive classes, so we fill both S and $l_{positive}$ into bitmap to remove duplicates.

4 EXPERIMENTS

To verify the effectiveness and efficiency of our ANN Softmax, we conduct experimental comparisons on two tasks: embedding learning and classification.

Embedding Learning: we try to verify the quality of feature extractor trained by various approaches. All models are trained on the refined version of MS-Celeb-1M (MS1MV2)[4], which consists of about 5.8M images and 85K classes. For validation, we have six testing datasets, including Labelled Faces in the Wild (LFW) [21], AgeDatabase (AgeDB) [32], Celebritiesin Frontal Profile (CFP) [33], Cross-Pose LFW (CPLFW) [50], Cross-Age LFW (CALFW) [51] and MegaFace[5] [23]. LFW is evaluated by face verification, whose metric is the verification accuracy via 10-fold cross validation. CFP, AgeDB, CPLFW and CALFW follow the same evaluation as LFW. For MegaFace, we record rank-1 identification accuracy with 1M distractors.

Classification: we choose ImageNet [7] and SKU datasets [35] with a varying number of classes including 1M, 10M, 100M for training and evaluation. And we add a large scale dataset SKU-300M with 300M classes, 4 billion training images and 0.5 billion testing images to evaluate the scalability of different methods. Top-1 accuracy is used as the metric of classification.

The state-of-the-art baselines and their hyper-parameters are listed below:

Full Softmax: Standard softmax using all classes for training.

HF-Softmax [45]: We use HF-A version with $L = 100, T = 1000$ and $\tau_{cp} = 0.9$.

KNN Softmax [35]: k is set to 6 for 1K, 12 for 1M, 120 for 10M and 1200 for 100M classes.

LSH-Softmax [26]: We set $b = \log_2 C$ and $L = 500$.

RP [44]: The random projection size m is set to 100.

Partial FC [2]: Sampling rate $r = 0.1$.

D-Softmax [20]: We use $\mathcal{L}_D$ loss; for D-Softmax-B, we sample 0.1 training samples from mini-batch; for D-Softmax-K, we sample 0.1 classes.

We use ResNet50 [8, 19] as backbone model in each experiment. For face recognition tasks, we use ArcFace [8] or CosFace [42] to enhance the discriminative power of feature embeddings learning. In ArcFace, the parameters of scale s and arccos margin m are set to 64 and 0.5 respectively. And the cosine margin m of CosFace is set to 0.4. For classification tasks, we use the standard cross entropy loss for training. The mini-batch size of MS1MV2 and ImageNet is set to 2048. We set the base learning rate to 0.4 for MS1MV2 and 0.8 for ImageNet training based on linear scaling rule [17, 24]. For MS1MV2, the learning rate is divided by 10 at 25K, 40K iterations and terminated at 45K iterations. When training ImageNet, we set the learning rate 10 times smaller at epoch 30, 60, and 80 while limit total training epochs to 90. For SKU datasets, we follow the fast convergence strategy [35] to adjust both learning rate and mini-batch size.

All of the experiments are running on a GPU cluster with 32 machines and each machine owns 8 Tesla V100-32G GPUs. We use PyTorch as the base framework and equip all methods with hybrid parallel training architecture and mixed precision (O1 version)[6] for fair comparison.

[4]https://github.com/deepinsight/insightface/wiki/Dataset-Zoo.

[5]The refined version used in ArcFace.

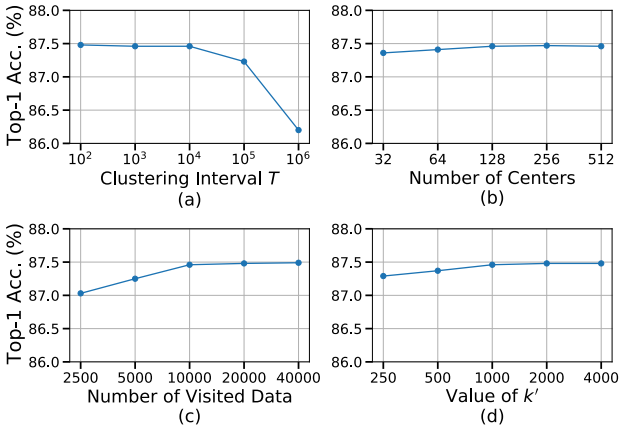

Figure 4: Performance vs. four parameters: T, $center_{num}$, HM_{num} and k'.

Table 1: The Top-1 Accuracy with different ANN recalls.

Method	ANN Softmax	HF-Softmax	LSH-Softmax
Recall / %	85.64	39.42	31.25
Accuracy / %	87.46	86.39	86.24

4.1 Ablation Study

The ablation study is performed on SKU-1M dataset. There are seven parameters in our ANN Softmax: C_{sub}, $center_{num}$, k', HM_{num}, T, g_{num} and k. We set $C_{sub} = \frac{1}{10}C$ for two reasons: 1. If C_{sub} is set too small, like 1/64, the selected active classes will be reduced, which may decrease training accuracy; 2. Setting C_{sub} to $\frac{1}{10}C$ can bring enough benefits since the computation and memory of FC layer/softmax loss become 1/10 of Full Softmax. The remaining six parameters are divided into the following two parts:

4.1.1 IVF-BQ. The four parameters: $center_{num}$, k', HM_{num} and T are closely related to the recall of IVF-BQ and the training accuracy. Figure 4 shows the influence of these parameters on the final accuracy. (a) shows the performance is not sensitive to clustering interval T. Only when T is greater than a threshold (= 10^6), there is a clear decline in accuracy. Because in that case, the centers will not be updated, so the result of IVF-BQ will be close to random sampling. Let each epoch have E iterations, T is ofter set to $1/5E$. The accuracy of different center numbers $center_{num}$ is almost the same

[6]https://nvidia.github.io/apex/.

Table 2: Comparisons of different k and g_{num}.

g_{num}	16	8	4	2	1
k	97	48	24	12	6
Accuracy / %	87.5	87.47	87.46	87.1	86.88

Table 3: Face recognition performance with different methods. The dotted line indicates baseline.

Method	LFW	CALFW	CPLFW	AgeDB	CFP	MegaF
Full Softmax	99.82	**96.18**	92.37	97.72	97.21	97.82
Partial FC	99.77	96.03	92.27	97.72	97.86	97.38
LSH-Softmax	99.77	96.05	92.28	97.73	97.80	97.40
HF-Softmax	99.79	96.08	92.30	97.72	97.79	97.44
RP	99.63	95.63	90.48	96.33	95.93	89.88
KNN Softmax	99.82	96.16	92.36	**97.74**	**97.90**	97.81
ANN Softmax	**99.83**	96.17	**92.39**	97.73	97.84	**97.82**

in (b), and the one with larger $center_{num}$ is slightly better than that with smaller $center_{num}$. Proportional to C, we choose $center_{num}$ from the range [64, 1024]. HM_{num} determines how many data will be visited, which is an important factor for the precision. As shown in (c), the larger HM_{num} is, the higher the accuracy will be. But too large HM_{num} will increase the ANN time. Considering both efficiency and accuracy, we set HM_{num} to $\frac{1}{10}C$ for ImageNet/SKU datasets and $\frac{1}{20}C$ for MS1MV2, respectively. In (d), k' has a similar behavior as $center_{num}$, and is usually set as $\frac{1}{10}HM_{num}$.

We compare three ANN-based sampling methods in Table 1 to display the relationship between ANN recall and accuracy. Obviously, accuracy is positively correlated with recall. ANN Softmax balances the traversal data of each sample and rerank binary results with float feature, so its recall is significantly higher than the other two methods. By the way, the above four parameters have different influences on the accuracy, because of their different effects on the recall. LSH-Softmax has the lowest accuracy, since the random projection has limited guarantee for recall.

4.1.2 Sample Grouping. Given C_{sub}, B and M, Eq (8) tells k is decided by g_{num}. We change g_{num} from 16 to 1, and show the performance in Table 2. It can be found that k or g_{num} has a great impact on the accuracy. The larger k is, the more active classes of each sample are recalled, leading to better approximation of Full Softmax and higher accuracy. And the accuracy of g_{num}=1 (no sample grouping) is lowest, proving our sample grouping optimization is effective. Setting g_{num} too large will increase the memory of $tempW$, so we often choose g_{num} to make the $global\ k\ of\ one\ sample$ (=$k \times M$) not greater than $\frac{N}{5000}$ in implementation.

4.2 Results on Embedding Learning Tasks

Next, we present the performance of different methods on embedding learning tasks in Table 3. All methods are combined with ArcFace to train the feature extractor model. It can be seen our ANN Softmax achieves nearly the same accuracy as Full Softmax on all datasets, proving its efficiency and stability. HF-Softmax and LSH-Softmax perform better than Partial FC since they use ANN strategy to recall more active classes, but they can not ensure most active classes are recalled, which may lead to insufficient training. The random sampling used by Partial FC ignores the importance of active class, resulting in worst performance among the sampled softmax baselines. RP doesn't perform well especially in CPLFW, AgeDB, CFP and MegaFace, the accuracy drop is more than 1%. Since it compresses the feature embeddings to a smaller space so that the ability of face representation gets damaged.

Table 4: ANN Softmax with various loss functions. No sampling ($r = 1$) occurs in ArcFace, CosFace and D-Softmax, others are $r = 0.1$.

Method	LFW	CALFW	CPLFW	AgeDB	CFP	MegaF
ArcFace	99.82	**96.18**	92.37	97.72	97.21	97.82
Ours ($r = 0.1$)	**99.83**	96.17	**92.39**	**97.73**	**97.84**	**97.82**
CosFace	99.70	95.93	**92.10**	97.83	97.46	98.00
Ours ($r = 0.1$)	**99.71**	**95.93**	92.09	**97.84**	**97.68**	**98.10**
D-Softmax	99.67	95.62	**91.23**	97.17	97.49	**96.78**
D-Softmax-B	**99.72**	95.63	90.77	97.18	97.00	96.71
D-Softmax-K	99.59	95.32	90.50	97.12	96.86	95.91
Ours*($r = 0.1$)	99.69	**95.65**	91.22	**97.18**	**97.50**	96.76

Table 4 shows the precision of our ANN Softmax combined with various loss functions on different face datasets. ArcFace and CosFace are widely used in face recognition community. D-Softmax is a new softmax variant which dissects softmax into independent intra-class and inter-class objectives to boost the feature expression. It has two light versions D-Softmax-B and D-Softmax-K for massive-scale training. D-Softmax-K samples classes, which can be easily integrated with our ANN Softmax (i.e., Ours* in Table 4). As expected, our method is comparable with baselines on all loss functions and datasets, exhibiting its robustness and extensibility.

4.3 Results on Classification Tasks

In Table 6, we show the classification results of different methods on ImageNet and SKU datasets. We vary the number of classes from 1K to 100M to further explore their limitations on extreme classification problem. It is very clear that our proposed method has the same performance as Full Softmax, and outperforms other methods on both datasets. Noted that as the number of classes increases, the precision loss of some methods like HF-Softmax, LSH-Softmax, RP and Partial FC become larger. For example, Partial FC has 0.28% accuracy loss in ImageNet, but the gap grows to 8.21% in SKU-100M. This can be attributed to the fact active classes will increase as the number of classes increases, so does the impact of active classes. Consequently, these methods that rarely or partly sample active classes will encounter more and more serious accuracy degradation when the number of classes get larger and larger. RP has higher accuracy than sampling-based methods, since it only performs feature compression, without any active classes reduction. The experimental results on both embedding learning and classification demonstrate that our sampling strategy: making each training data recall large quantity and good quality of active classes, is more reasonable.

Table 5: The time profiling of different methods in three large-scale SKU datasets. FC_{fwd} means the time in forward of FC layer, $loss_{fwd}$ is the time in backward of softmax loss, and *graph* stands for the building time of graph in KNN Softmax.

Method	r	N	M	B	Time profiling (ms/iter)						Mem (GB)	Top-1 Acc of SKU (%)		
					FC_{fwd}	$loss_{fwd}$	$loss_{bwd}$	FC_{bwd}	graph	Total		10M	100M	300M
Full Softmax	1	10M	64	128	26.12	74.31	48.48	33.26	/	319	9.68	81.01	/	/
KNN Softmax	0.1	10M	64	128	6.47	14.25	4.78	7.97	41.87	202	8.01	80.99	/	/
ANN Softmax	0.1	10M	64	128	19.2	11.88	4.3	13.06	/	**162**	**7.83**	81.00	/	/
Full Softmax	1	100M	128	54	109.12	316.28	158.61	183.00	/	964	31.4	/	74.52	/
KNN Softmax	0.1	100M	128	54	23.44	27.92	16.27	35.79	70.13	276	26.11	/	74.54	/
ANN Softmax	0.1	100M	128	54	45.05	27.57	16.69	43.21	/	**224**	**22.86**	/	74.53	/
Full Softmax	1	300M	256	32	/	/	/	/	/	/	OOM	/	/	/
KNN Softmax	0.1	300M	256	32	/	/	/	/	/	/	OOM	/	/	/
ANN Softmax	0.1	300M	256	32	90.85	58.71	40.36	106.73	/	**394**	**31.5**	/	/	70.45

Table 6: Image classification performance with different methods.

Method	ImageNet	1M	10M	100M
Full Softmax	76.50	87.43	**81.01**	74.52
Partial FC	76.22	86.15	76.73	66.31
LSH-Softmax	76.29	86.24	78.16	69.83
HF-Softmax	76.31	86.39	79.02	71.98
RP	75.11	87.16	79.83	72.19
KNN Softmax	76.49	87.46	80.99	**74.54**
ANN Softmax	**76.56**	**87.49**	81.00	74.53

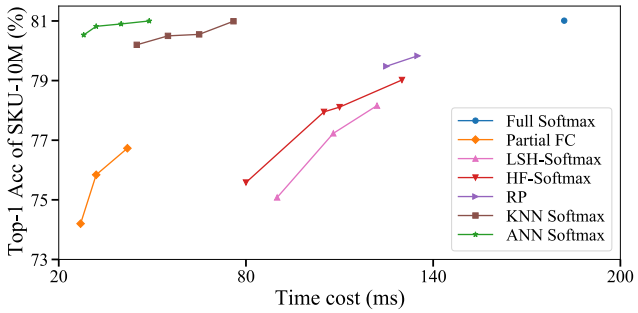

Figure 5: Accuracy vs. Time cost of different methods.

4.4 Time Profiling

As depicted in Figure 5, we plot the time-accuracy curves of all methods on SKU-10M dataset. As we mentioned above, the time spent on FC layer and softmax loss occupies the majority of each iteration, so we focus on their time consumption. We use the Top-1 accuracy with the time cost from FC layer and softmax loss as a metric, so the points that near the top-left corner indicate high performance with low cost. The results show that: (1) All methods can reduce the computation cost. But those totally implemented by GPU, such as ANN Softmax, KNN Softmax and Partial FC, are apparently faster than others, like HF-Softmax and LSH-Softmax. (2) Without any ANN procedure, Partial FC is the fastest, but the accuracy is also reduced heavily, which is not acceptable in practical applications. Even if we improve its sample rate to make its speed same as ANN Softmax, the accuracy is still below ours. (3) RP is another option to solve massive classification problem. Leave the loss of accuracy aside, it only reduces feature dimensions without any classes sampling, which means no acceleration in softmax loss stage. And recovering the final weights of classifier will introduce more training time compared with other methods. (4) Our ANN Softmax can reduce the computation time up to 270% and maintain lossless accuracy, which is a remarkable improvement in practical scenarios.

At last, we display the time profiling results on three large-scale SKU datasets in Table 5. For fair comparison, we only choose the methods that keep the same precision with Full Softmax, i.e., KNN Softmax and ANN Softmax. And we add the graph building time (divided into each iteration) for KNN Softmax, which can not be ignored in training process. Obviously, our ANN Softmax is fastest no matter on SKU-10M or 100M. We achieve 4.3× speedup than Full Softmax, and 1.23× speedup than KNN Softmax, respectively.

Although KNN Softmax keeps almost the same performance as our method in previous experiments, it runs slower than ANN Softmax. What's worse, the resources spent on graph construction and storage are impracticable when N is very large, like 300M, limiting its scalability. Thanks to 256 Tesla V100-32G equipped with ANN Softmax, we succeed in training a 300M SKU classifier within five epochs and ten days. And the accuracy 70.45% is acceptable in online application.

5 CONCLUSIONS

In this paper, we regard the quantity and quality of active classes recalled by one training sample as the key to maintain the same accuracy with Full Softmax. In particular, we propose IVF-BQ to do the active classes selection, and parallelize it with well designed CUDA kernel function. Then we introduce two kinds of sample grouping optimization to improve the sampled active classes quantity of each training sample. Experiments on a variety of tasks, datasets and loss functions demonstrate that our ANN Softmax is lossless in accuracy with only 1/10 sampled classes. What's more, our method has successfully trained a classifier of 300 million classes on 256 GPUs cluster in ten days, which sets up a new state-of-the-art for the community.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation (OSDI)*. 265–283.

[2] Xiang An, Xuhan Zhu, Yang Xiao, Lan Wu, Ming Zhang, Yuan Gao, Bin Qin, Debing Zhang, and Ying Fu. 2020. Partial FC: Training 10 Million Identities on a Single Machine. *arXiv preprint arXiv:2010.05222* (2020).

[3] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2016. Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. In *International Conference on Very Large Data Bases (VLDB)*. 288–299.

[4] Akhil Arora, Sakshi Sinha, Piyush Kumar, and Arnab Bhattacharya. 2018. HD-index: pushing the scalability-accuracy boundary for approximate kNN search in high-dimensional spaces. In *International Conference on Very Large Data Bases (VLDB)*. 906–919.

[5] Jiajiong Cao, Yingming Li, and Zhongfei Zhang. 2018. Celeb-500k: A large training dataset for face recognition. In *IEEE Conference on Image Processing (ICIP)*. 2406–2410.

[6] Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906* (2015).

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4690–4699.

[9] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: table understanding through representation learning. In *International Conference on Very Large Data Bases (VLDB)*. 307–319.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 4171–4186.

[11] Matthijs Douze, Hervé Jégou, and Florent Perronnin. 2016. Polysemous codes. In *European Conference on Computer Vision (ECCV)*. 785–801.

[12] Matthijs Douze, Alexandre Sablayrolles, and Hervé Jégou. 2018. Link and code: Fast indexing with graphs and compact regression codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3646–3654.

[13] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. In *International Conference on Very Large Data Bases (VLDB)*. 461–474.

[14] Jinyang Gao, Hosagrahar Visvesvaraya Jagadish, Wei Lu, and Beng Chin Ooi. 2014. DSH: data sensitive hashing for high-dimensional k-nnsearch. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1127–1138.

[15] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *International Conference on Very Large Data Bases (VLDB)*. 518–529.

[16] Joshua Goodman. 2001. Classes for fast maximum entropy training. In *IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1. 561–564.

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[18] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 297–304.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[20] Lanqing He, Zhongdao Wang, Yali Li, and Shengjin Wang. 2020. Softmax Dissection: Towards Understanding Intra-and Inter-class Objective for Embedding Learning. In *The AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 34. 10957–10964.

[21] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report. University of Massachusetts, Amherst.

[22] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33, 1, 117–128.

[23] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. 2016. The megaface benchmark: 1 million faces for recognition at scale. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4873–4882.

[24] Alexandros Koliousis, Pijika Watcharapichat, Matthias Weidlich, Luo Mai, Paolo Costa, and Peter Pietzuch. 2019. Crossbow: scaling deep learning with small batch sizes on multi-GPU servers. In *International Conference on Very Large Data Bases (VLDB)*. 1399–1412.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)* 25, 1097–1105.

[26] Daniel Levy, Danlu Chen, and Stefano Ermon. 2017. LSH Softmax: Sub-Linear Learning and Inference of the Softmax Layer in Deep Architectures. In *Workshop of Advances in Neural Information Processing Systems (NeurIPSW)*.

[27] Jinfeng Li, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Deep or simple models for semantic tagging? it depends on your data. In *International Conference on Very Large Data Bases (VLDB)*. 2549–2562.

[28] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. In *International Conference on Very Large Data Bases (VLDB)*.

[29] Qiang Long, Wei Wang, Jinfu Deng, Song Liu, Wenhao Huang, Fangying Chen, and Sifan Liu. 2019. A distributed system for large-scale n-gram language models at Tencent. In *International Conference on Very Large Data Bases (VLDB)*. 2206–2217.

[30] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. 2007. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *International Conference on Very Large Data Bases (VLDB)*. 950–961.

[31] Tharun Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. 2019. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. In *Advances in Neural Information Processing Systems (NeurIPS)*. 13244–13254.

[32] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. 2017. Agedb: the first manually collected, in-the-wild age database. In *Workshop of the IEEE Conference on Computer Vision and Pattern Recognition (CVPRW)*. 51–59.

[33] C.D. Castillo V.M. Patel R. Chellappa D.W. Jacobs S. Sengupta, J.C. Cheng. 2016. Frontal to Profile Face Verification in the Wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

[34] Wissam Siblini, Pascale Kuntz, and Frank Meyer. 2019. A review on dimensionality reduction for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

[35] Liuyihan Song, Pan Pan, Kang Zhao, Hao Yang, Yiming Chen, Yingya Zhang, Yinghui Xu, and Rong Jin. 2020. Large-Scale Training System for 100-Million Classification at Alibaba. In *ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD)*. 2909–2930.

[36] Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. 2014. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. In *International Conference on Very Large Data Bases (VLDB)*. 1529–1540.

[37] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020. Circle loss: A unified perspective of pair similarity optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6398–6407.

[38] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. 2020. ODIN: Automated Drift Detection and Recovery in Video Analytics. In *International Conference on Very Large Data Bases (VLDB)*. 2453–2465.

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3104–3112.

[40] Sudheendra Vijayanarasimhan, Jonathon Shlens, Rajat Monga, and Jay Yagnik. 2015. Deep networks with large output spaces. In *Workshop of International Conference on Learning Representations (ICLRW)*.

[41] Guanhua Wang, Zhuang Liu, Brandon Hsieh, Siyuan Zhuang, Joseph Gonzalez, Trevor Darrell, and Ion Stoica. 2021. sensAI: ConvNets Decomposition via Class Parallelism for Fast Inference on Live Data. *Proceedings of Machine Learning and Systems (MLSys)* 3 (2021).

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

[43] Xiao-Lin Wang, Hai Zhao, and Bao-Liang Lu. 2013. A meta-top-down method for large-scale hierarchical classification. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 26, 3, 500–513.

[44] Zhuoning Yuan, Zhishuai Guo, Xiaotian Yu, Xiaoyu Wang, and Tianbao Yang. 2020. Accelerating Deep Learning with Millions of Classes. In *European Conference on Computer Vision (ECCV)*. 711–726.

[45] Xingcheng Zhang, Lei Yang, Junjie Yan, and Dahua Lin. 2018. Accelerated training for massive classification via dynamic class selection. In *The AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 32.

[46] Yuhao Zhang and Arun Kumar. 2019. Panorama: a data system for unbounded vocabulary querying over video. In *International Conference on Very Large Data Bases (VLDB)*. 477–491.

[47] Yanhao Zhang, Pan Pan, Yun Zheng, Kang Zhao, Yingya Zhang, Xiaofeng Ren, and Rong Jin. 2018. Visual search at alibaba. In *ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD)*. 993–1001.

[48] Kang Zhao, Hongtao Lu, and Jincheng Mei. 2014. Locality preserving hashing. In *The AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 28.
[49] Kang Zhao, Pan Pan, Yun Zheng, Yanhao Zhang, Changxu Wang, Yingya Zhang, Yinghui Xu, and Rong Jin. 2019. Large-Scale Visual Search with Binary Distributed Graph at Alibaba. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 2567–2575.
[50] Tianyue Zheng and Weihong Deng. 2018. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing University of Posts and Telecommunications, Tech. Rep* 5 (2018).
[51] Tianyue Zheng, Weihong Deng, and Jiani Hu. 2017. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv preprint arXiv:1708.08197* (2017).