Accelerating Deep Learning with Millions of Classes

Zhuoning Yuan[1], Zhishuai Guo[1], Xiaotian Yu, Xiaoyu Wang[2], and Tianbao Yang[1]

[1] The University of Iowa, Iowa City, IA 52242, USA
[2] The Chinese University of Hong Kong (Shenzhen), Shenzhen, China
{zhuoning-yuan, zhishuai-guo, tianbao-yang}@uiowa.edu

Abstract. Deep learning has achieved remarkable success in many classification tasks because of its great power of representation learning for complex data. However, it remains challenging when extending to classification tasks with millions of classes. Previous studies are focused on solving this problem in a distributed fashion or using a sampling-based approach to reduce the computational cost caused by the softmax layer. However, these approaches still need high GPU memory in order to work with large models and it is non-trivial to extend them to parallel settings. To address these issues, we propose an efficient training framework to handle extreme classification tasks based on *Random Projection*. The key idea is that we first train a slimmed model with a random projected softmax classifier and then we recover it to the original version. We also show a theoretical guarantee that this recovered classifier can approximate the original classifier with a small error. Later, we extend our framework to parallel scenarios by adopting a communication reduction technique. In our experiment, we demonstrate that the proposed framework is able to train deep learning models with millions of classes and achieve above 10× speedup compared to existing approaches.

Keywords: large-scale learning, deep learning, random projection, acceleration in deep learning

1 Introduction

One of the biggest advantages of deep learning is to improve the capability of modeling complex data over conventional approaches. In practice, deep learning has achieved the state-of-the-art performance on various applications in computer vision and natural language processing [12,8]. With the dramatic increase of scale in data, extreme classification emerges as a challenging task. This leads to some new challenges: (1) the extremely large model size, i.e., the model size is dominated by softmax layer, which is proportional to the number of classes; (2) the expensive cost to train such models. A widely used benchmark dataset, ImageNet [19], contains only 1000 classes. However, for applications such as facial recognition and language modeling, the number of classes can easily reach

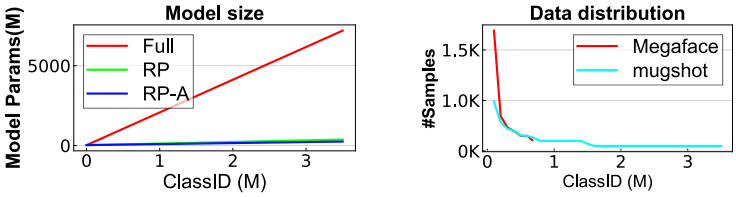

Fig. 1. Illustrations of model size by varying the number of classes and the class distribution of two facial datasets. Fig. 1 shows the size of model with full softmax increases dramatically as the increase of number of classes compared to slimmed model with projected softmax. Fig 2. reveals that facial datasets follow long-tailed distribution.

a million-scale, e.g., Megaface [18] and Google Billion Words [5]. To reduce the cost of training, we utilize a well-known dimensionality reduction technique, *Random Projection*, e.g., a high-dimensional feature $\mathbf{x}$ can be reduced to a low-dimensional one through a random matrix R. To understand the idea behind the random projection, the key result is built on Johnson-Lindenstrauss lemma [2], which reveals that any high dimensional data can be mapped onto a lower-dimensional space while the distance between data points is approximately persevered. Compared to other reduction methods. e.g., SVD, random projection has a lower computational cost and thus it is being widely used in image and text tasks [3]. We summarize the current progress of accelerating training for extreme classification in the following section.

1.1 Naive Methods

Typically, a single machine is not enough to handle a large-scale task efficiently and therefore the most straightforward solution is to directly increase the number of workers and distribute the task over these machines. As a result, it leads to an immediate improvement of training efficiency. For example, a distributed system of 256 GPUs can train ResNet50 on ImageNet with batch size of 8192 in one hour with a top1 validation accuracy of 75.7% [9]. Recently, it has been extended to a larger system of 2176 Tesla V100 GPUs, which takes only 224 seconds for training to achieve a similar performance [22]. Considering the trade-off between model precision and training efficiency, mixed precision training [16] is another solution to handle the large-scale tasks with a lower memory demand. The key idea is to convert the model weights, activations and gradients to lower precision, e.g., float16, during forward and backward pass, but a copy of weights in float32 are maintained during parameter updates. In addition, it is also possible to distribute the computation process of softmax layer based on matrix partition to multiple parallel workers in order to overcome the difficulty of training large models on a large-scale dataset [7].

1.2 Sampling-based Methods

In the applications of Natural Language Processing, there are two common ways to tackle extreme classification: sampling-based and softmax-based approaches. These works are generally inspired by an important fact, Zipf's law, which tells that there are a small number of vocabularies (classes) with high occurrence and a very large number of vocabularies (classes) with low occurrence in the dataset. This rule also applies to many facial datasets, e.g., Celeb and MegaFace as shown in figure 1. For sampling-based methods, Zhang et al. [27] identified that only $1\% \sim 2\%$ classes are active during training on facial recognition task. Accordingly, they proposed an algorithm dynamically selecting those active classes to speed up the training. Their experiments show a significant reduction on GPU memory cost and an improvement of training efficiency. The key idea of softmax-based approaches is to convert a multi-class task to a binary task by sampling a small set of negative samples along with the positive sample according to Zipf's law, e.g., noise-contrastive estimation and sampled softmax [1,11]. In addition, some works [10,18] indicate that pretraining on a small subset sampled from the original dataset by class frequency is also a useful sampling-based strategy to speed up model converge compared to training from scratch for large models.

1.3 Challenges

We summarize the existing challenges to tackle the extreme classification from two perspectives as follows:

1. **Memory limitations**. As the increase of model size, the GPU memory demand to train such model grows rapidly as illustrated in figure 1. Sampling-based approaches can reduce this cost by computing sampled softmax scores, but the model size remains unchanged. Although we can apply additional compression techniques to reduce model size, it may also suffer a performance drop.
2. **Communication limitations**. To extend the training to parallel settings, if we naively increase the number of workers in the system, the communication cost among these workers can be a new bottleneck for the training. Unfortunately, the most existing approaches fail to consider this scenario and are unlikely to be deployed in practice and obtain a linear speedup for parallel training.

1.4 Our Contributions

The main contributions of our work to tackle the above issues are summarized as follows:

1. We propose a two-stage algorithm to tackle extreme classification. We first solve a slimmed problem with random projected softmax classifier to avoid the heavy computational cost of original problem with full softmax and then we recover the projected classifier to its original version.

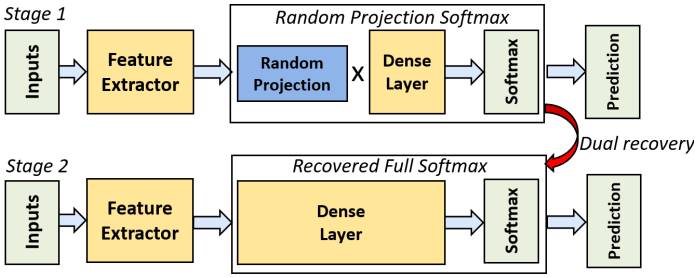

Fig. 2. The training framework of random projected softmax. In stage one, we reduce full softmax to projected softmax to speed up the training. In the second stage, we finetune the recovered softmax layer to further boost the performance.

2. We show a theoretical analysis that our recovered classifier can achieve a relative small error compared to original classifier. In addition, we verify that the embedding features extracted from the slimmed model are as good as the features from the original model.
3. We present a feasible solution with communication reduction technique to extend the proposed framework to parallel settings and we show a nearly linear speed-up of our algorithm with respect to the number of workers as well as much less communication cost.
4. We conduct a comprehensive empirical study to verify our algorithm on different scales of datasets, and we also investigate the important factors that may affect the performance by various ablation studies.

2 Proposed Approach

In this section, we formally introduce our two-stage proposed framework as described in Algorithm 1 and illustrated in Figure 2. At the first stage, we aim to quickly solve the slimmed problem with random projected softmax layer to avoid the heavy computational cost when the number of classes is large. At the second stage, we propose a dual recovery approach to recover the projected softmax classifier to its original version. In addition, we adopt a communication reduction technique to further accelerate the training in parallel settings. We present the details of these approaches in the following sections.

2.1 Solving Problem with Projected Softmax Layer

Firstly, we formulate the slimmed problem as

$$\min_{\mathbf{z}} \mathcal{L}(\mathbf{z}) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{z}; \mathbf{x}_i, y_i), \tag{1}$$

where $\mathcal{L}(\mathbf{z}; \mathbf{x}_i, y_i)$ is a loss function with a regularization term, $(\mathbf{x}_i, y_i)$ is a pair of feature and label for a random sample i out of n samples, and $\mathbf{z}^*$ is the solution

Algorithm 1 A Two-stage framework for extreme classification

input: input X, Y, $\mathbf{z}_0$, and m
for s=1 to S **do**
  $\mathbf{z}_s$ =PSGD($\mathbf{z}_{s-1}, \eta_s, T_s, I_s$) {S is number of stages }
end for
Compute the dual solution $\widehat{\Theta}$ using $\mathbf{z}_S$ by **prop. 1**,
$$\widehat{\Theta} = \left(\nabla L_{y_1}\left(\hat{\mathbf{o}}_1\right), \ldots, \nabla L_{y_n}\left(\hat{\mathbf{o}}_n\right)\right)$$
Compute recovered classifier $\widetilde{W} \in \mathbb{R}^{d\times K}$ by $\widetilde{W} = -\frac{1}{n\lambda} X\widehat{\Theta}$
output: $\widetilde{\mathbf{W}}$

of the problem (1), including the feature extractor $f(\mathbf{x})$ and the corresponding projected softmax classifier $\widehat{W}$. Using a Gaussian random matrix $R^{d\times m}$, where d is the embedding size and m is the random projection size, we generate a low-dimensional representation for each input feature $f(\mathbf{x}_i)$ by

$$\hat{f}(\mathbf{x}_i) = \frac{1}{\sqrt{m}} R^\top f(\mathbf{x}_i) \tag{2}$$

We specify $\widehat{W} = (\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_C) \in \mathbb{R}^{m\times C}$, where C is the number of classes. For a feature extractor $f(\cdot)$ and random projection matrix R, to learn the projected softmax classifier can be formulated as follows,

$$\min_{\widehat{W}} \frac{\lambda}{2}\|\widehat{W}\|_F^2 + \sum_{i=1}^{n} \ell_{y_i}\left(\hat{\mathbf{o}}_i\right) \tag{3}$$

where $\hat{\mathbf{o}}_i = (\hat{\mathbf{o}}_{i,1}, \ldots, \hat{\mathbf{o}}_{i,C}) = \left(\hat{\mathbf{w}}_1^\top \hat{f}(\mathbf{x}_i), \hat{\mathbf{w}}_2^\top \hat{f}(\mathbf{x}_i), \ldots, \hat{\mathbf{w}}_C^\top \hat{f}(\mathbf{x}_i)\right)$. Since the feature extractor and the projected classifier are trained together, we could suppose that the learned projected classifier is optimal to the learned features.

Remarks: When C is sufficiently large, e.g., $C = 10^6$, solving the original model can be computational expensive and time consuming. Instead, solving slimmed model with projected softmax layer is more efficient. In addition, we argue that the learned embedding features from the slimmed model are as good as the embedding features from original model and we verify this in the experiments.

2.2 Solving Recovery Problem

Denote $W \in \mathbb{R}^{d\times C}$ as the original softmax classifier without projection corresponding to feature extractor $f(\mathbf{x})$. Noting that the size of $\widehat{W}$ is much less than the size of W, it is expected that the original classifier can produce better predictions than the projected classifier. Therefore, in the second stage, we recover W from $\widehat{W}$ by fixing feature extractor $f(\mathbf{x})$. In this setting, the problem can be formulated as follow,

$$\min_{W} \frac{\lambda}{2}\|W\|_F^2 + \sum_{i=1}^{n} \ell_{y_i}\left(\mathbf{o}_i\right), \tag{4}$$

Algorithm 2 PSGD: Proximal Stochastic Gradient Descent $(\mathbf{z}_0, \eta, T, I)$

1: **for** $t = 1$ to T **do**
2: Each machine k updates its local solution in parallel:
3: $\mathbf{z}_{t+1}^k = \mathbf{z}_t^k - \eta(\nabla\psi(\mathbf{z}_t^k; \xi_t^k) + \frac{1}{\gamma}(\mathbf{z}_t^k - \mathbf{z}_0))$
4: **if** $t + 1 \bmod I = 0$ **then**
5: $\mathbf{z}_{t+1}^k = \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_{t+1}^k$ {K is total number of workers}
6: **end if**
7: **end for**

where $\mathbf{o}_i = (\mathbf{o}_{i,1}, \ldots, \mathbf{o}_{i,C}) = \left(\mathbf{w}_1^\top f(\mathbf{x}_i), \ldots, \mathbf{w}_C^\top f(\mathbf{x}_i)\right)$ and λ is a parameter for the regularized term. Note that the problem (3) is a projected version of the problem (4). Next, we show that we can recover the projected softmax classifier to the original softmax classifier with a small error for any feature extractor $f(\cdot)$.

Recall that $f(\mathbf{x})$ is the embedding features of the data $\mathbf{x}_i$ generated by the network. $\mathbf{o}_i$ is the scores of $\mathbf{x}_i$ over C classes, and $\mathbf{o}_i = (\mathbf{o}_{i,1}, \ldots, \mathbf{o}_{i,C}) = \left(\mathbf{w}_1^\top f(\mathbf{x}_i), \ldots, \mathbf{w}_C^\top f(\mathbf{x}_i)\right)^T = \mathbf{W}^\top f(\mathbf{x}_i)$. For any i, we write $\ell_{y_i}(\mathbf{o}_i)$ in its conjugate form as

$$\ell_{y_i}(\mathbf{o}_i) = \max_{\boldsymbol{\theta}_i}[\mathbf{o}_i^T\boldsymbol{\theta}_i - \ell_{y_i}^*(\boldsymbol{\theta}_i)] \tag{5}$$

Plugging (5) to (4), we get the dual optimization problem of (4)

$$\max_{\boldsymbol{\Theta}} -\sum_{i=1}^{n} \ell_{y_i}^*(\boldsymbol{\theta}_i) - \frac{1}{2\lambda}\|\mathbf{X}\Theta^{*T}\|_F^2, \tag{6}$$

where $\Theta = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n)$. We denote $W^* \in \mathbb{R}^{d\times C}$ as the optimal primal solution to (4), and denote $\Theta^* \in \mathbb{R}^{C\times n}$ as the optimal dual solution to (6) and the following proposition connects W^* and Θ^*.

Proposition 1. *Let $W^* \in \mathbb{R}^{d\times C}$ be the optimal primal solution to (4) and $\boldsymbol{\Theta}^* \in \mathbb{R}^n$ be the optimal dual solution to (6). We have*

$$W^* = -\frac{1}{\lambda}\mathbf{X}\Theta^{*T}, \text{ and } \Theta^* = (\nabla L_{y_1}(\mathbf{o}_1^*), \ldots, \nabla L_{y_n}(\mathbf{o}_n^*)), \tag{7}$$

where $\mathbf{o}_i^* = W^{*T}\mathbf{x}_i$, $i = 1, ..., n$.

Similarly, we have the following proposition for primal and dual solution of the projected softmax classifier.

Proposition 2. *Let $\widehat{W}^* \in \mathbb{R}^{m\times C}$ be the optimal primal solution of the projected problem and $\hat{\Theta}^* \in \mathbb{R}^{n\times C}$ be the optimal dual solution for the projected problem. We have*

$$\widehat{W}^* = -\frac{1}{\lambda}\frac{1}{m}R^T X\widehat{\Theta}^{*T}, \text{ and } \widehat{\Theta}^* = (\nabla L_{y_1}(\hat{\mathbf{o}}_1^*), \ldots, \nabla L_{y_n}(\hat{\mathbf{o}}_n^*)), \tag{8}$$

where $\hat{\mathbf{o}}_i^* = \frac{1}{m}(\mathbf{R}\mathbf{z}_i^*)^T\mathbf{x}_i, \ldots, \frac{1}{m}(\mathbf{R}\mathbf{z}_C^*)^T\mathbf{x}_i$, $i = 1, ..., n$.

In the following theorem, we show that we can recover $\widehat{W}^*$ to W^* with a small error.

Theorem 1. *We recover the optimal solution from dual variables* $\widetilde{W} = -\frac{1}{\lambda}\mathbf{X}\widehat{\Theta}^{*T}$. *For any* $0 < \epsilon \leq 1/2$, *with a probability at least* $1 - \delta$, *we have*

$$\|\widetilde{W} - W^*\|_F \leq \frac{\epsilon}{1-\epsilon}\|W^*\|_F, \tag{9}$$

provided

$$m \geq \frac{(r+1)\log(2r/\delta)}{c_0\epsilon^2},$$

where constant c_0 *is at least* $1/4$, *and* r *is the rank of* $f(X) = f(\mathbf{x}_1, ..., \mathbf{x}_n)$.

Remarks: The above theorem shows that the recovery error between recovered softmax classifier and original softmax classifier is bounded with a relative small error, which implies that we can use the recovered softmax classifier $\widetilde{W}$ to approximate the original W. The proof is skipped and included in the supplement.

2.3 Parallel Training with Communication Reduction

For deep learning in practice, the scales of model and dataset can easily reach a million or billion levels, which makes it very inefficient to train on a single GPU. To further accelerate the training, a parallel multi-worker system is a common solution. In this setting, each worker performs an individual task concurrently and updates the local solution by the average of all individuals (e.g., gradient or weight averaging) at each iteration. For the moderate scale classification task, the averaging cost is usually ignored compared to other costs. However, this is not the case for extreme classification. The communication cost of averaging can be a real bottleneck for overall training process. To address this issue, we adopt a strategy inspired by *Parallel Restarted SGD* [23] to reduce the number of communication rounds between local workers and the gradient algorithm with the communication reduction is presented in Algorithm 2.

Here, we use the same formulation as in the first stage of our approach:

$$\min_{\mathbf{z}} \mathcal{L}(\mathbf{z}) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{z}; \mathbf{x}_i, y_i), \tag{10}$$

where $\mathcal{L}(\mathbf{z}; \mathbf{x}_i, y_i)$ is a loss function. Next, we analyze the convergence rate of $\mathcal{L}(\mathbf{z})$ for Algorithm 1 with Algorithm 2 and we show how distributed machines can speed up the training and how can we reduce the communication cost.

We make the following assumptions for analysis:

Assumption 1
(1) Loss function $\mathcal{L}(\mathbf{z}; \xi)$ *is* L*-smooth on* $\mathbf{z}$ *for any* i. *Thus it is also* L*-weakly convex.*
(2) $\|\nabla\mathcal{L}(\mathbf{z}) - \nabla\mathcal{L}(\mathbf{z}; \xi)\|_2^2 \leq \sigma^2$ *for any* i.
(3) $\mathcal{L}(\mathbf{z})$ *satisfies* μ*-PL condition, i.e.,* $\mu(\mathcal{L}(\mathbf{z}) - \mathcal{L}(\mathbf{z}^*_{\mathcal{L}})) \leq \frac{1}{2}\|\nabla\mathcal{L}(\mathbf{z})\|^2$.

The analysis of one call of Algorithm 2 is shown in the following lemma.

Lemma 1 (Analysis of Algorithm 2). *Suppose Assumption 1 holds and* $\|\nabla\mathcal{L}(\mathbf{z};\mathbf{x}_i,y_i)+\frac{1}{\gamma}(\mathbf{z}-\mathbf{z}_0)\|_2 \leq B$. *Let* $\psi(\mathbf{z}) = \sum_{i=1}^{n}\mathcal{L}(\mathbf{z};\mathbf{x}_i,y_i)+\frac{1}{2\gamma}\|\mathbf{z}-\mathbf{z}_0\|^2$. *By running Algorithm 2, taking* $\gamma=\frac{1}{2L}$, *and* $\eta\leq\frac{1}{3L}$, *we have*

$$E[\mathcal{L}(\mathbf{z})-\mathcal{L}(\mathbf{z}_*)] \leq \frac{\|\mathbf{z}_0-\mathbf{z}^*\|^2}{2\eta T}+2L\eta^2I^2B^2\mathbb{I}_{I>1}+\frac{\eta\sigma^2}{K}.$$

With this lemma, we have the following theorem to show the convergence of the Algorithm 1.

Theorem 2. *Suppose the same condition in Lemma 2 holds. Set* $\gamma=\frac{1}{2L}$, $\eta_0\leq\frac{1}{3LK}$ *and* $c_1=\frac{\mu/L}{5+\mu/L}$. *Take* $\eta_s=\eta_0K\exp(-(s-1)c_1)$, $T_s=\frac{2}{L\eta_0K}\exp\left((s-1)c_1\right)$ *and* $I_s=\max(1,\frac{1}{\sqrt{K\eta_s}})$. *To return* $\bar{\mathbf{z}}_S$ *such that* $E[\mathcal{L}(\bar{\mathbf{z}}_S)-\mathcal{L}(\mathbf{z}_*)]\leq\epsilon$, *the number of iterations is at most* $\widetilde{O}\left(\max(\frac{1}{\mu\epsilon\eta_0K},\frac{1}{\mu^2K\epsilon})\right)$ *and the number of communications is at most* $\widetilde{O}\Big(\max\left(\frac{K}{\mu}+\frac{1}{\mu(\eta_0\epsilon)^{1/2}},\frac{K}{\mu}+\frac{1}{\mu^{3/2}\epsilon^{1/2}}\right)\Big)$, *where* $\widetilde{O}$ *suppresses logarithmic factors and appropriate constants.*

Remark. Take $\eta_0=\frac{1}{3LK}$. If $K\leq O(\frac{1}{\mu})$, then then number of iteration is dominated by $O(\frac{L}{\mu^2K\epsilon})$ since μ is very small in deep learning [24]. Then we can see that there is a linear speedup over the convergence speed with respect to the number of machines K. In addition, the number of communications is much less than the number of total iterations. When $I=1$, algorithm 2 reduces to standard parallel SGD.

3 Experiments

3.1 Implementation Details

Tasks. We conduct a comprehensive study to verify the effectiveness of our proposed methods on two tasks. **Representation Learning**: we aim to justify the quality of the feature extractor trained on slimmed model with projected softmax classifier and the quality of the feature extractor from the same model under different parallel settings. The validation process is evaluated on a facial verification task using only embedding features and thus the recovery producers are ignored. We implement our algorithms on three facial datasets, Celeb [10], Megaface [18] and Mugshot [6]. For validation, we test our models on three benchmark sets, Labelled Faces in the Wild (LFW), AgeDatabase (AgeDB) and Celebritiesin Frontal Profile (CFP). **Classification**: we aim to justify the quality of the recovered softmax classifier and the training and evaluation are both done on classification datasets. Specifically, we use ImageNet [22], Penn Treebank (PTB) [20] and Google Billion Words (GBW) [5] with a varying number of

Table 1. Datasets Summary. "M" and "K" indicate millions and thousands.

datasets	#samples/words	#classes	task
ImageNet	1.2 M	1 K	classification
Penn Treebank	929 K	10 K	classification
Google Billion Words	800 M	0.79 M	classification
MS-Celeb-1M	5.3 M	93 K	representation
MegaFace2	4.7 M	0.67 M	representation
Mugshot-1M	3.5 M	1 M	representation

classes from one thousand to one million. The detailed descriptions of the above datasets are summarized in Table 1.

Experiment setup. For the preprocessing steps, we follow the similar producers of their original works for all datasets. For ImageNet, we apply random cropping and left-right flipping on raw images, resulting in 224×224 random images at each iteration. For facial datasets, we augment the images by left-right flipping and resize them to 112×112. For language modeling datasets, we reshape the data by time steps in order to fit the time-series model (LSTM). The facial and language datasets are sorted by the class occurrence in order to implement sampling-based methods. We adopt the ResNet50 and two variants of LSTM as backbones and we use multiple 11GB GTX2080-Ti or a 32 GB V100 for training.

Parameters. We employ the original version of ResNet50 [13] with the embedding size of 2048 for all images classification tasks. The baselines are trained using SGD and L2 regularization with a weight decay of 0.0001. The learning rate is scheduled in a stage-wise fashion: starting with a initial learning rate 0.1, dividing it by 10 at {30, 60, 90} and {10, 16} epoch for ImageNet and Facial datasets [7]. For language modeling tasks, we use the configurations of LSTM [14] from [25,17]. For GBW, Adagrad with initial learning rate of 0.2 is used for optimization. The size of word embedding and feature embedding are {1500,1500} and {1024,1024} for PTB and GBW, respectively. The number of time steps for LSTM model is set to 35 and 20 respectively. The initial random seeds are fixed for all experiments.

Evaluation Metrics. We use *GPU Hrs* to measure the training speed for all experiments. The speedup is defined as T_2/T_1 in order to measure the relative improvement of the first algorithm with respect to the second algorithm, where T_1 is the runtime for the first algorithm and T_2 is the runtime for the second algorithm. For representation learning, we report *verification accuracy* for face verification task, identifying whether two given face images are the same person. For language modeling, we evaluate *perplexity*, i.e., $2^{-\sum_x p(x) \log_2 p(x)}$, where $p(x)$ is the output probability of a sample x.

Table 2. Model performance varying the size of random projection m.

RP(m)	Celeb	Megaface	Mugshot	Imagenet	RP(m)	PTB	GBW
10	0.934	0.815	0.631	0.696	10	137.59	122.16
100	0.965	0.845	0.671	0.740	100	86.15	52.28
1000	0.960	0.831	0.664	0.739	**500**	83.98	46.34

3.2 Ablation Study on the Choice of Random Projection Size

We first explore the effect of the model performance by varying the random projection size m. Considering both model performance and training efficiency, we set the upper bound of projection size m to be the half of embedding size d. The embedding sizes of ResNet50/LSTM are $\{2048, 1500, 1024\}$, thus we set $m = \{10, 100, 500/1000\}$. For facial datasets, we report the average score of three validation sets. Table 2 shows the model behaviors of different choice of m for all datasets. The results tell that the best choice of m is 100 for Celeb, MegaFace and Mugshot. The best choice for language datasets is $m = 500$ with the lowest test perplexity. We fix $m = 100/500$ for the following experiments.

3.3 Ablation Study on Multi-GPU Scaling Efficiency

We further explore the runtime by varying different number of classes in a 8-GPU GTX2080-Ti workstation. We choose ResNet50 as our main network which has around 23.5 million parameters. We compare the full softmax (Full) with the random projected (RP) softmax, e.g., $m = 100$ under different number of communication reduction rounds $I = \{4, 8, 32\}$. We fit the simulated images ($112 \times 112 \times 3$) to GPU memory before training to avoid the penitential issues caused by data I/O operations. We choose number of classes C to be {10K, 100K, 1000K, 3500K}, leading to model size: Full = {43.98M, 228.30M, 2071M, 7801M}, RP ={24.5M, 33.5M, 123.5M, 345M}. Figure 3 shows the scaling efficiency over the number of GPUs used. We can see that the cost of communication (averaging operations) can be ignored when the number of classes C is small and we can obtain a significant performance boost by increasing the number of GPUs. However, when C reaches million level, vanilla Full and RP softmax have terrible scaling performance and RP softmax with communication reduction I has a clear advantage. By choosing a proper I, RP softmax can even achieve nearly linear speedup when $C = 3.5$ millions.

3.4 Ablation Study on Communication Rounds

From the results in previous section, we realized that both Full softmax and RP softmax ($I = 1$) have limits when C is large. Next, we investigate the proper choice of communication round I in order to achieve a good balance in both training speed and model performance. We first vary I by fixing K (#gpus). We compare $I = \{1, 4, 8, 32\}$ on $K = \{2, 4, 8\}$. We conduct the experiments on MegaFace, which contains 0.67M classes and all models are trained with the

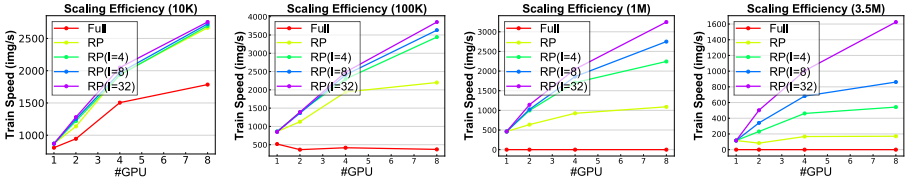

Fig. 3. Multi-GPU scaling efficiency on different settings. The RP softmax has a clear advantage over Full softmax when number of classes is over 100K.

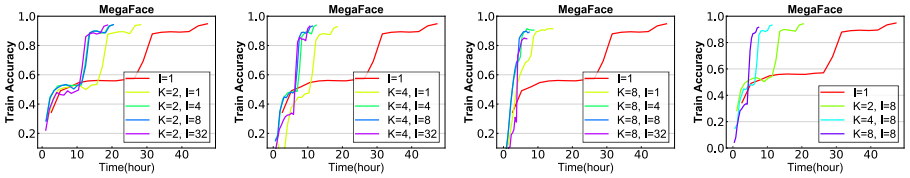

Fig. 4. Performance of different choice of I and K. For the first three figures, we fix K and vary I. For the last figure, we fix I and vary K.

same number of epochs. In figure 4, $I = 1$ ($K = 1$) is the baseline trained on a single GPU with batch size of 256. As the increase of number K, we also use the increasing batch size $K \times 256$ with initial learning rate $K \times \eta_0$. From the results, we found that the algorithms with $I >= 1$ generally coverage much faster than baseline. In particular, $I = 8$ seems to be the best choice considering both speed and performance. However, we noticed that there is a significant performance drop when $I = 32$, which may indicate the upper limits of the choice of I. In the last figure, we show a parallel result by varying K and fixing I.

4 Evaluation Results

We compare our proposed approach with several baselines. The full description of all methods is listed below:

1. **Full-D**: full softmax with data parallel
2. **Full-D-C**: full softmax with data and center parallel
3. **S-S**: sampled softmax [15]
4. **Pretrain**: pretrained model on a random sampled subset of full dataset
5. **RP**: random projected softmax classifier
6. **RP-A**: an adaptive variant of random projected softmax classifier

4.1 Results on Speedup

We first examine the results of the training speed (GPU hour) of listed methods. We set a fixed batch size for all experiments on the same dataset, leading to the different number of utilized GPUs due to the varying model size. The detailed

Table 3. Training efficiency and speedup

	Celeb-1M	MegaFace	Mugshot-1M	ImageNet	PTB	GBW
Full-D	14.18 (1.00×)	NA	NA	2.38 (1.00×)	0.061(1.00×)	3.51 (1.00×)
Full-D-C	4.96 (2.86×)	27.36 (1.00×)	28.96 (1.00×)	2.36 (1.00×)	NA	NA
S-S	4.69 (3.02×)	NA	NA	2.36 (1.00×)	0.061(1.00×)	0.53 (6.62×)
Pretrain	NA	NA	NA	NA	NA	NA
RP	**2.39 (5.93×)**	**2.53 (10.83×)**	**2.94 (11.84×)**	2.36 (1.00×)	0.055(1.11×)	**0.26 (12.5×)**
RP-A	2.60 (5.46×)	2.72 (10.03×)	3.32 (10.48×)	NA	0.055(1.11×)	1.32 (2.66×)

Table 4. Model performance on validation sets

	Celeb-1M			MegaFace			Mugshot-1M			ImageNet	PTB	GBW
	LFW	CFP	AgeDB	LFW	CFP	AgeDB	LFW	CFP	AgeDB	Val acc	Perplexity	Perplexity
Full-D	99.55	97.03	94.97	NA	NA	NA	NA	NA	NA	**75.69**	78.26	59.07
Full-D-C	99.57	**97.34**	95.03	95.58	79.63	71.25	84.33	66.60	50.82	75.12	NA	NA
S-S	99.03	94.10	88.05	NA	NA	NA	NA	NA	NA	70.11	79.99	47.48
Pretrain	98.91	87.80	92.53	NA	NA	NA	NA	NA	NA	NA	NA	NA
RP	**99.62**	96.80	**95.22**	**96.60**	**82.46**	**74.40**	**90.70**	**69.71**	**57.27**	73.98	83.98	**46.38**
Recover	NA	NA	NA	NA	NA	NA	NA	NA	NA	74.56	**74.78**	46.56
RP-A	99.48	97.14	95.42	96.95	83.99	74.85	90.83	68.97	59.23	NA	81.76	74.3

evaluations are presented in Table 3. For the pair of two numbers in the table, the first number denotes GPU hour per epoch and the second number denotes the speedup. We set the speedup for the baseline to **1.00** for the convenience. NA denotes the method is not applicable due to memory issues or other reasons. From the results, we can see that as the increase in the number of classes, RP softmax approaches achieve significant improvement for training efficiency on facial recognition tasks. This is reasonable since we directly reduce the model parameters on softmax layer. Unlike ResNet50, RP softmax on LSTM achieves a lower speedup on a similar scale dataset, e.g., MegaFace v.s. GBW. This is due to the difference between two network architectures: ResNet has 23.5M while LSTM has 450M parameters for the model excluding the softmax layer. Thus, we combine the RP softmax with sampled softmax, which achieves a speedup of 12.5× v.s. 2.97× without sampled softmax.

4.2 Results on Model Performance

Representation Learning. Here, our focus is to verify the quality of the feature extractor by slimmed model with projected softmax classifier. We report the verification accuracy based on embedding features on three sets in table 4. From the table, we observe some counter-intuitive results that RP-based approaches outperform full softmax baseline in general. Recall our observations in the figure 1 that there is a dramatic increase in the model size for full softmax model, this potentially matches another fact that the extremely large models are more difficult to train than slimmed models under the limited choice of step size, batch size, number of epochs. This explains why RP-based methods achieve better performance. Meanwhile, the results also verify our previous argument that the slimmed model can also generate high-quality embedding features as the original model. In terms of training speed, RP-based methods spend much less time on

training than original baselines in overall.

Classification. We compute the classification scores for evaluation on validation sets. From the results in table 4, there are several key observations: 1) RP softmax achieves better score than S-S softmax in GBW while there is no performance boost on PTB. 2) Training with RP softmax on ImageNet has no improvement on both performance and speed. The first observation can be explained by a similar reason that the extremely large model is more difficult to train but we can actually improve this score by recovery approach. The second observation on ImageNet is due to the parameters in softmax classifier only takes a very small portion of entire model and thus projected softmax will not lead to a significant parameter reduction and improvement on training speed. In addition, ImageNet doesn't follow the long-tailed distribution, and thus random projected softmax may carry some negative effect on the model performance.

4.3 Results on Recovered Softmax Classifier

From previous section, we noticed that training with random projected softmax suffers a performance drop on the datasets with a small number of classes. Thus, we investigate whether our recovery approach can recover the projected softmax classifier $\hat{\mathbf{W}}$ to full softmax classifier $\mathbf{W}$. We use pretrained model from stage 1 to recover the full classifier $\mathbf{W}$ from $m \times C$ to $d \times C$. In this case, $L_{y_i}(\mathbf{o}_i) = -\log \frac{\exp([\mathbf{o}_i]_{y_i})}{\sum_{k=1}^{C} \exp([\mathbf{o}_i]_k)}$, the recovery formula is shown as follow

$$\widetilde{W} = -\frac{1}{\lambda}\mathbf{X}(P(\widehat{W}^*) - \mathbf{Y})^T \tag{11}$$

where $P(\widehat{W}^*) \in \mathbb{R}^{n \times C}$ indicates the predicted probability by $\widehat{W}^*$, i.e., $[P(\widehat{W}^*)]_{i,j} = \frac{\exp \mathbf{w}_i^{*T}\mathbf{x}_j}{\sum_{c=1}^{C} \exp \mathbf{w}_c^{*T}\mathbf{x}_j}$, and $\mathbf{Y} \in \{0,1\}^{C \times n}$ denotes the labels in one-hot encoding. Then, we finetine the recovered softmax classifier for some epochs. In table 4, the recovered classifier improves 1% accuracy after 4 epochs training on ImageNet, which is slightly worse than baseline. For PTB, we achieve a test perplexity of 78.29 after training for 18 epochs and achieve a perplexity of 74.78 after training for 33 epochs. For GBW, we train 10 hours to recover the projected classifier to achieve a competitive test perplexity of 46.56. The results indicate that the recovered classifier has a similar or even better decision-making capability compared to original classifier.

4.4 Results on Parallel Training with Communication Reduction

We explore the optimal choice of I and K to accelerate the parallel training on Mugshot and Celeb datasets. We denote the proposed algorithm 2 as RP with model averaging **(RP-model)** and compare it with three baselines, Full-D, Full-D-C, RP with gradient averaging **(RP-grad)**. We compare $K = \{2, 4, 8\}$ and

Table 5. Model Performance on Parallel Training.

Methods	K	I	Mugshot			Train Time		K	I	Celeb			Train Time	
			LFW	CFP	AgeDB	GPU	Hrs			LFW	CFP	AgeDB	GPU	Hrs
Full-D	8	NA	NA	NA	NA	NA	NA	4	1	99.57	97.34	95.03	14.18	3.55
Full-D-C	8	1	84.33	66.60	50.82	28.96	3.62	4	1	99.55	97.03	94.97	4.96	1.24
RP-grad	2	1	90.70	69.71	57.27	2.94	1.47	2	1	99.62	96.68	95.22	2.39	1.20
RP-model	2	4	90.75	68.77	59.23	1.95	0.98	2	4	99.70	97.45	95.91	2.04	1.02
RP-model	2	8	91.03	70.57	61.05	1.90	0.95	2	8	99.60	97.07	95.46	2.08	1.04
RP-model	4	4	90.72	69.14	60.96	2.42	0.61	4	4	99.68	97.25	96.00	2.47	0.62
RP-model	4	8	89.80	67.21	59.93	2.09	0.52	4	8	99.65	97.34	95.78	2.35	0.59
RP-model	8	4	88.20	67.65	59.61	3.47	0.43	8	4	99.66	97.25	95.76	3.29	0.41
RP-model	8	8	85.45	67.20	56.23	2.83	0.35	8	8	99.60	97.11	95.05	3.12	0.39

$I = \{4, 8\}$ and use the increasing batch size and learning rate of all experiments for each dataset. The results are reported in table 5 in terms of verification accuracy and training time. It shows that RP-model outperforms the baselines when $K = 2$ on Mugshot and Celeb. Further, as we increase K to 8, training time per epoch can be reduced by a large margin of 90%, e.g., from 3.62 to 0.35 hrs when $I = 8$ on Mugshot and from 3.55 to 0.39 hrs when $I = 8$ on Celeb. In other words, we can train one million classification task in just few hours instead of days for Full-D. However, we also observe some performance drops when I, K are relatively large, but the results are still competitive to Full-D/Full-D-C.

4.5 An Adaptive Variant of Random Projected Softmax

Inspired by [1,27], we propose an adaptive variant of RP softmax, called adaptive RP softmax (RP-A). The intuition is simple: we assign a larger projection size to "head" classes and a smaller projection size to "tail" classes since "head" classes contain more information than "tail" classes. To implement it, we group all classes into clusters according to their occurrence, and assign each cluster with a varying projection size m from large to small. For example, we group all classes into four clusters with $m = \{100, 70, 40, 10\}$ in our experiments. As the results in table 4, RP-A outperforms the baselines in some settings.

5 Conclusion

In this paper, we proposed an effective framework to tackle extreme classification problem. Our framework is able to train deep learning models with millions of classes on various tasks. In our numerical experiments, we have verified that the slimmed model with projected softmax classifier can also generate high-quality embedding features as good as embedding features trained from original model and meanwhile our methods spend less time on training. In addition, we also demonstrate that the recovered softmax classifier can achieve a competitive or even better classification performance. Finally, we successfully extend our framework to large-scale parallel settings and also obtain good results in terms of both training efficiency and model performance.

References

1. Blanc, G., Rendle, S.: Adaptive sampled softmax with kernel based sampling. arXiv preprint arXiv:1712.00527 (2017)
2. Borwein, J., Lewis, A.S.: Convex analysis and nonlinear optimization: theory and examples. Springer Science & Business Media (2010)
3. Boutsidis, C., Zouzias, A., Drineas, P.: Random projections for k-means clustering. In: Advances in Neural Information Processing Systems. pp. 298–306 (2010)
4. Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge university press (2006)
5. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., Robinson, T.: One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005 (2013)
6. Data.gov: Nist mugshot identification database mid. NIST Mugshot Identification Database MID - NIST Special Database 18 (2016)
7. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2019)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
10. Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: European Conference on Computer Vision. pp. 87–102. Springer (2016)
11. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 297–304 (2010)
12. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
15. Jean, S., Cho, K., Memisevic, R., Bengio, Y.: On using very large target vocabulary for neural machine translation. arXiv preprint arXiv:1412.2007 (2014)
16. Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L., et al.: Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. arXiv preprint arXiv:1807.11205 (2018)
17. Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016)
18. Kemelmacher-Shlizerman, I., Seitz, S.M., Miller, D., Brossard, E.: The megaface benchmark: 1 million faces for recognition at scale. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4873–4882 (2016)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

20. Miltsakaki, E., Prasad, R., Joshi, A.K., Webber, B.L.: The penn discourse treebank. In: LREC (2004)
21. Nesterov, Y.E.: Introductory Lectures on Convex Optimization - A Basic Course, Applied Optimization, vol. 87. Springer (2004)
22. You, Y., Zhang, Z., Hsieh, C.J., Demmel, J., Keutzer, K.: Imagenet training in minutes. In: Proceedings of the 47th International Conference on Parallel Processing. p. 1. ACM (2018)
23. Yu, H., Yang, S., Zhu, S.: Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 5693–5700 (2019)
24. Yuan, Z., Yan, Y., Jin, R., Yang, T.: Stagewise training accelerates convergence of testing error over SGD. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. pp. 2604–2614 (2019)
25. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
26. Zhang, L., Mahdavi, M., Jin, R., Yang, T., Zhu, S.: Recovering the optimal solution by dual random projection. In: Conference on Learning Theory. pp. 135–157 (2013)
27. Zhang, X., Yang, L., Yan, J., Lin, D.: Accelerated training for massive classification via dynamic class selection. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)