Paper 1682: Supplementary Material
Large-Scale Distributed Learning via Private On-Device LSH

A Experiment details

In this section, we provide deeper background into how our experiments were run as well as some additional results and observations. We first detail the hyper-parameters we used in order to reproduce our results. Then, we provide additional comments and details into our sampling approach. Finally, we describe some of the interesting observations we encountered while solving the Amazon-670K and Wiki-325K recommender system problems.

A.1 Experiment hyper-parameters

Below, we detail the hyper-parameters we used when running our federated experiments.

Table 1: **Hyper-parameters for Federated Experiments (PGHash and Federated SLIDE).**

Dataset	Algorithm	Hash Type	LR	Batch Size	Steps per LSH	k	c	Tables	CR
Delicious-200K	PGHash	PGHash	1e-4	128	1	8	8	50	1
Delicious-200K	SLIDE	SimHash	1e-4	128	1	8	N/A	50	1
Amazon-670K	PGHash	PGHash-D	1e-4	256	50	8	8	50	1
Amazon-670K	SLIDE	DWTA	1e-4	256	50	8	N/A	50	1
Wiki-325K	PGHash	PGHash-D	1e-4	256	50	5	16	50	1
Wiki-325K	SLIDE	DWTA	1e-4	256	50	5	N/A	50	1

What one can immediately see from Table 1, is that we use a Densified Winner Take All (DWTA) variant of PGHash for the larger output datasets Amazon-670K and Wiki-325K. As experienced in [8, 7, 24], SimHash fails to perform well on these larger datasets. We surmise that SimHash fails due in part to its inability to select a large enough number of neurons per sample (we observed this dearth of activated neurons empirically). Reducing the hash length k does increase the number of neurons selected, however this decreases the accuracy. Therefore, DWTA is used because it utilizes more neurons per sample on these larger problems and also still achieves good accuracy.

Table 2: **Hyper-parameters for Compression Experiments (PGHash).**

Dataset	Algorithm	Hash Type	LR	Batch Size	Steps per LSH	k	c	Tables	CR
Delicious-200K	PGHash	PGHash	1e-4	128	1	8	8	50	0.1/0.25/1

As a quick note, we record test accuracy every so often (around 100 iterations for Delicious-200K and Amazon-670K). Similar to [8], to reduce the test accuracy computations (as the test sets are very large) we compute the test accuracy of 30 randomly sampled large batches of test data.

A.2 Neuron sampling

Speed of Neuron Sampling. In Table 3 we display the time it takes to perform LSH for PGHash given a set number of tables. These times were collected locally during training. The entries in Table 3 denote the time it takes to compute hashing of the final layer weights w_i and each sample x in batch M as well as vanilla-style matching (neuron selection) for each sample.

Table 3: **Average LSH time for PGHash over a range of tables.** We compute the average μ time (and standard deviation σ) it takes for PGHash to perform *vanilla sampling* (exact matches) between the hash codes of sample x and each weight w_i in the final dense layer. Times are sampled for PGHash on Delicious-200k for batch size $M = 128$, $k = 9$, and $c = 8$ for one device.

Method	1 table (seconds)	50 tables (seconds)	100 tables (seconds)
PGHash	$\mu = 0.0807, \sigma = 0.0076$	$\mu = 3.1113, \sigma = 0.0555$	$\mu = 6.2091, \sigma = 0.1642$
SLIDE	$\mu = 0.0825, \sigma = 0.0099$	$\mu = 3.2443, \sigma = 0.1671$	$\mu = 6.2944, \sigma = 0.0689$

We find in Table 3 that PGHash achieves near sub-linear speed with respect to the number of tables τ and slightly outperforms SLIDE. PGHash edges out SLIDE due to the smaller matrix multiplication

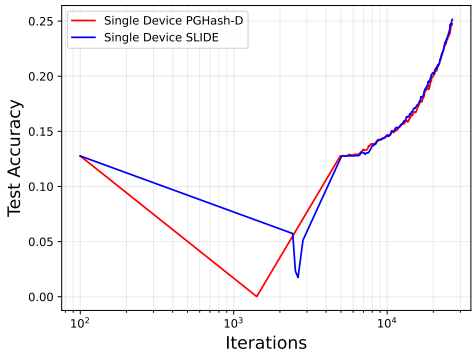


Figure 7: **Wiki-325K PGHash-D.** We record model accuracy of a large recommendation system on Wiki-325K. PGHash-D matches the convergence of SLIDE without requiring LSH to be performed by the central server. We note that test accuracy is determined by testing 30 randomly sampled large batches of test data (and not the full test data). We saw that the true full test accuracy (which we compute after each epoch) ran about 5% greater than the sampled batches.

cost, as PGHash utilizes a smaller random Gaussian matrix (size $c \times c$). The speed-up over SLIDE will become more significant when the input layer is larger (as $d = 128$ in our experiments). Therefore, PGHash obtains superior sampling performance to SLIDE.

Hamming Distance Sampling. An alternative method to vanilla sampling is to instead select final layer weights (neurons) w_i which have a small Hamming distance relative to a given sample x. As a refresher, the Hamming distance simply computes the number of non-matching entries between two binary codes (strings). If two binary codes match exactly, then the Hamming distance is zero. In this sampling routine, either **(i)** the top-k weights w_i with the smallest Hamming distance to sample x are selected to be activated or **(ii)** all weights w_i with a Hamming distance of β or smaller to sample x are selected to be activated. Interestingly, the vanilla-sampling approach we use in our work is equivalent to using $\beta = 0$ in **(ii)**.

In either of the scenarios listed above, hash codes for w_i and x are computed as done in PGHash(-D). From there, however, the hash code for x is compared to the hash codes for all final layer weights in order to compute the Hamming distance for each w_i. The process of computing n Hamming distances for each sample x is very expensive (much harder than just finding exact matches). That is why our work, as well as [8, 7], use vanilla sampling instead of other methods.

A.3 Amazon-670K and Wiki-325K experiment analysis

Sub-par SimHash Performance. SimHash is known to perform worse than DWTA on Amazon-670K and Wiki-325K. Utilizing SimHash for these experiments is unfair as it is shown by [8, 7], for example, that DWTA achieves much higher performance on Amazon-670K. For this reason, DWTA is the chosen hash function in [8] for Amazon-670K experiments. To verify this observation, we performed experiments on Amazon-670K with PGHash (not PGHash-D) and SLIDE (with a SimHash hash function). Table 4 displays the SimHash approach for Amazon-670K.

Table 4: **PGHash and SLIDE performance on Amazon-670K using SimHash.** Accuracy across the first 5,000 iterations for a single device. Batch size $M = 1024$, $k = 8$, and $c = 8$.

Iteration	SLIDE	PGHash
1,000	10.82%	10.04%
2,000	18.27%	15.99%
3,000	21.83%	19.51%
4,000	23.72%	21.65%
5,000	25.08%	23.38%

As shown in Table 4, even with a much larger batch size, SLIDE and PGHash are unable to crack 30% on Amazon-670K. We would like to note that using a smaller batch size (like the $M = 256$

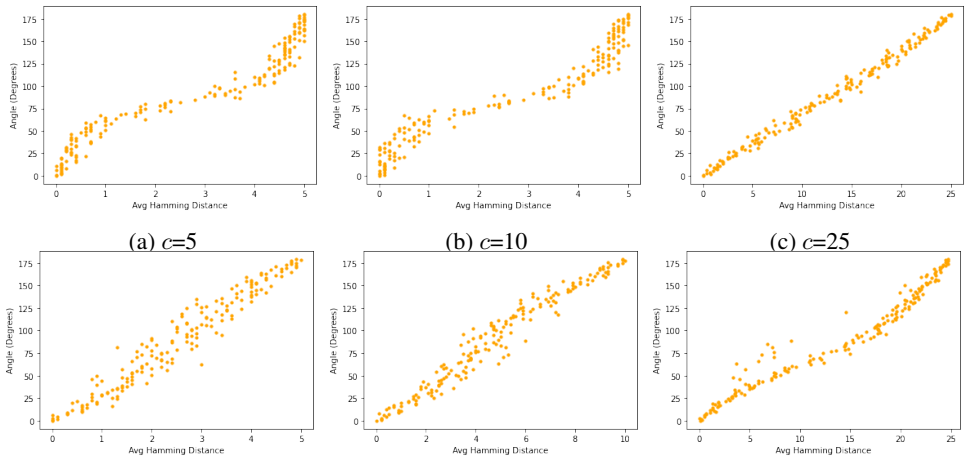
value we use in our Amazon-670K experiments) resulted in an even further drop in accuracy. These empirical results back-up the notion that SimHash is ill-fit for Amazon-670K.

Wiki-325K Performance. In Figure 7, we showcase how PGHash-D performs on Wiki-325K. Quite similar to the Amazon-670K results (shown in Figure 5), PGHash-D almost exactly matches up with SLIDE. In order to map how well our training progresses, we periodically check test accuracies. However, since the test set is very large, determining test accuracies over the entire test set is infeasible due to time constraints on the cluster. Therefore, we determine test accuracies over 30 batches of test data as a substitute as is done in [8, 7]. For Delicious-200K and Amazon-670K the entire test set accuracies matched the randomly sampled batches, however the randomly sampled batches underestimate the true test accuracies for Wiki-325K. For Wiki-325K, the true test accuracy ran about 5% greater than the sampled test accuracy values.

Matching Full-Training Performance. Along with the failure for SimHash to perform well on Amazon-670K and Wiki-325K, SLIDE and PGHash(-D) are unable to match the performance of full-training on these data-sets. This is observed empirically for Amazon-670K by GResearch in the following article `https://www.gresearch.co.uk/blog/article/implementing-slide/`. We surmise that the failure of SLIDE and PGHash(-D) to match full-training performance on Amazon-670K and Wiki-325K arises due to the small average labels per point in these two data-sets (5.45 and 3.19 respectively). Early on in training, SLIDE and PGHash(-D) do not utilize enough activated neurons. This is detrimental to performance when there are only a few labels per sample, as the neurons corresponding to the true label are rarely selected at the beginning of training (and these final layer weights are tuned much slower). In full-training, the true neurons are always selected and therefore the final layer weights are better adjusted from the beginning. We also note that [33] requires a hidden layer size of 1024 for a distributed version of SLIDE to achieve improved test accuracies for Amazon-670K. Thus, increasing the hidden layer size may have improved our performance (we kept it as 128 to match the original SLIDE paper [8]).

B PGHash: angle versus Hamming distance

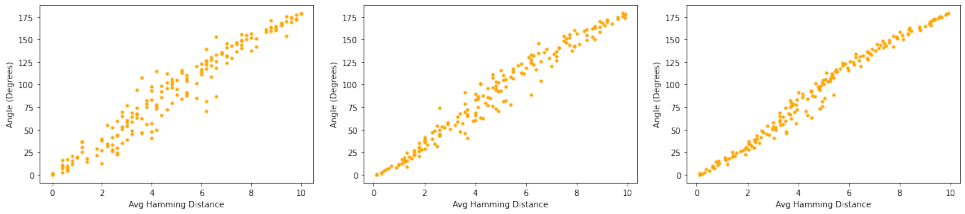
In this section, we visually explore the degree to which PGHash is a consistent estimator of angular similarity. Specifically, let $x, y \in \mathbb{R}^d$: then we know by Theorem 1 that $\mathcal{H}^{PG}(c, d)$ is an LSH for $\cos(x_c, y_c)$. We demonstrate that in the unit vector regime, $\theta_c = \arccos(\cos(x_c, Y_c))$ is an acceptable surrogate for $\theta = \arccos(x, Y)$, where $Y = \{y^i\}_{i=1}^N$ and $Y_c = \{y_c^i\}_{i=1}^N$.



(a) c=5 (b) c=10 (c) c=25

(d) c=5 (e) c=10 (f) c=25

Figure 8: **Angle/Hamming Distance as a function of sketch dimension.** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^{100}$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x. Increasing sketch dimension c smooths and reduces the variance of the scatter towards linear correlation. Furthermore, the Hamming scales linearly with c, improving discernibility. (a)-(c) & (d)-(f) are independent series.



(a) 5 tables (b) 10 tables (c) 50 tables

Figure 9: **Angle/Hamming Distance as a function of tables** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^100$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x and fixed sketch dimension $c = 10$. Increasing the number of tables reduces variance.

C Additional proofs

Fact 3. *Let x, y, e_1, e_2 be d-dimensional unit vectors such that that the e_i lie on the unit circle contained with the plane spanned by x and y (denoted as $S_{x,y}$) and $e_1 \perp e_2$. Consider the point v on $S_{x,y}$ such that the line through it bisects the angle of the lines passing through x and y. Let $\eta = \arccos(\cos(v, e_1)$. Denote $\theta = \frac{1}{2}\arccos(\cos(x, y))$. Then we may write $x = \cos(\eta + \theta)e_1 + \sin(\eta + \theta)e_2$ and $y = \cos(\eta - \theta)e_1 + \sin(\eta - \theta)e_2$.*

C.1 Proof of Theorem 2

Proof. Let $\theta = \frac{1}{2}\arccos(\cos(x, y))$ where $x, y \in \mathbb{S}^{d-1}$ and $A = B^\top B$. The cosine similarity between $x_c = Bx$ and $y_c = By$ (for B correspondent to a (d, c)-folding), is expressible as

$$\cos(x_c, y_c) = \frac{x^\top B^\top By}{(x^\top B^\top Bx)(y^\top B^\top By)} = \frac{x^\top Ay}{\sqrt{(x^\top Ax)(y^\top Ay)}}. \tag{1}$$

Consider the SVD $B = UDV^\top$ where U and V are orthogonal and D is $c \times d$ rectangular diagonal matrix. We have then that $A = B^\top B = V\hat{D}^2V^\top$. (Here $\hat{D}$ is now a square diagonal matrix containing squared D_{ii} along the diagonal and 0 everywhere else.) Notice that choice of U nor the ordering of columns v_i of V affects the angle calculation in Equation 1. First, we re-order the columns of V so as to order the diagonal entries d_i of D (i.e., the squared singular values) in decreasing order, and as an abuse of notation set $B = \frac{1}{d_1}DV^\top$. Denoting $\hat{\lambda}_i = d_i/d_1$ for $1 \leq i \leq n$, we have that $Bv_i = \hat{\lambda}_i e_i$. (By construction of B we have that $d_i \in \{\frac{d}{c}, 0\}$, therefore, $\hat{\lambda}_i \in \{1, 0\}$)

Consider B acting on S^{d-1}: it scales each dimension by $\hat{\lambda}_i$, thus (as with any linear transformation of a sphere), transforms it into an ellipsoid, with c principal axes determined by the v_i. The greatest possible distance from the origin to the ellipsoid BS^{d-1} is 1 while the shortest possible distance is 0. Now consider the unit circle $S_{x,y} = \{v \in \text{span}(x, y) \ : ||v|| = 1\}$. We have that $BS_{x,y} \subset BS^{d-1} \cap BU$ is an ellipse (since the intersection of an ellipsoid and plane is always an ellipse).

Choose unit w_1 and w_2 belonging to $S_{x,y}$ such that $w_1 \perp w_2$. by By Fact 3, we may parameterize our vectors as $x = \cos(\eta - \theta)w_1 + \sin(\eta - \theta)w_2$ and $y = \sin(\eta + \theta)w_1 + \sin(\eta + \theta)w_2$, where η is the angle made with w_1 with the bisector of x and y. By assumption, $||Bw|| \geq \alpha$ (the minimal shrinking factor of B on $S_{x,y}$), so denoting $\lambda = \frac{d}{c}$ (the maximal stretching factor of B on $S_{x,y}$), we have that the angle between Bx and By is upper-bounded by

$$f(\eta) = \arctan(\frac{\alpha}{\lambda}\tan(\eta + \theta)) - \arctan(\frac{\alpha}{\lambda}\tan(\eta - \theta)) \tag{2}$$

.

The numerator of $\frac{df}{d\eta}$ is $\beta(1 - \beta)(1 + \beta)\sin(2\theta)\sin(2\eta)$ where $\beta = \alpha/\lambda$. The derivative is trivially 0 if (1) $\beta = 0$, (2) $\beta = 1$, or (3) $\theta = 0$. (1) will not occur as we assume that $S_{x,y}$ does not contain a 0-eigenvector of $A = B^\top B$. (2) can only occur if A is a multiple of the identity matrix (which it is not by construction), and (3) implies that x and y are parallel, in which case their angle will not be

distorted. Aside from these pathological cases, the critical points occur at $\eta = 0, \pi/2$. We have then that $\cos(Bx, By)$ lives between $\cos(f(0)) = \frac{1-\beta^2 \tan^2 \theta}{1+\beta^2 \tan^2 \theta}$ and $\cos(f(\pi/2)) = -\frac{\tan^2 \theta^2 - \beta^2}{\tan^2 \theta^2 + \beta^2}$.

□

Remark. The constant β has an enormous influence on the bounds in Theorem 2. The smaller the α (i.e., shrinking of $||w||$), the greater the bounds on distortion. Although we have imposed constraints on x, y, if we treat them as any possible pair of random unit vectors, then the w in $S_{x,y}$ effectively becomes a random unit vector as well. We can exactly characterize the distribution of $||BX||$ where X denotes a random variable which selects a d-dimensional unit vector uniformly at random.

C.2 Proof of Proposition 1

Proof. We can sample a d-dimensional vector uniformly at random from the unit sphere S^{d-1} by drawing a d-dimensional Gaussian vector with iid entries and normalizing. Let us represent this as the random variable $X = Z'/||Z'||$ where $Z' \sim \mathcal{N}(0, I_d)$. Consider a (c, d)-folding matrix B, i.e., a d/c horizontal stack of $c \times c$ identity matrices (let us assume $c|d$). We are interested in determining the distribution of $||BX||^2$. For ease of notation, consider the permutation Z of Z' where $Z_i = Z'_{(\lfloor \frac{d/c}{i} \rfloor - 1)*(d/c)+i \pmod{d/c}}$. Since this permutation is representable as an orthogonal matrix P (and multi-variate Gaussians are invariant in distribution under orthogonal transformations), we may instead consider $X := P(Z'/||Z'||)^2 = Z/||Z||^2$. We may write the norm-squared as

$$||BX||^2 = \frac{(Z_1 + \cdots + Z_{d/c})^2}{||Z||^2} + \frac{(Z_{d/c+1} + \cdots + Z_{2d/c})^2}{||Z||^2} + \cdots + \frac{(Z_{(c-1)(d/c)+1} + \cdots + Z_d)^2}{||Z||^2}. \tag{3}$$

Consider the first term $\frac{(Z_1+\cdots+Z_{d/c})^2}{||Z||^2}$. First note that for any unit vector u, the distribution of $\frac{(u^\top Z)^2}{||Z||^2}$ does not depend on choice of u. Consider the unit vector u' then which contains $\sqrt{d/c}$ in the first d/c entries and 0 otherwise. Then $\frac{(u'^\top Z)^2}{||Z||^2}$ is equivalent to d/c times our first term. Of course, since $\frac{(e_1^\top Z)^2}{||Z||^2}$ has the same distribution as $\frac{(u'^\top Z)^2}{||Z||^2}$, we have by transitivity that $\frac{Z_1^2}{||Z||^2} \overset{d}{=} (n/q)\frac{(Z_1+\cdots+Z_{d/c})^2}{||Z||^2}$.

By extending the discussion above to the other terms, and by their independence with respect to rotation of Z (since their numerators contain squared sums of mutually disjoint Z coordinates), we have that

$$||BX||^2 \overset{d}{=} \frac{d}{c} \cdot \frac{Z_1^2 + Z_{d/c}^2 + Z_{2d/c}^2 + \cdots + Z_d^2}{||Z||^2}. \tag{4}$$

The distribution of $\frac{Z_1^2+Z_{d/c}^2+Z_{2d/c}^2+\cdots+Z_d^2}{||Z||^2}$ is well-known to follow a $\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2})$ distribution [13]. In totality, $||BX||^2 \overset{d}{=} \frac{d}{c}\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2})$. However, we will move to the four parameter description of this scaled Beta distribution which is $\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2}, 0, \frac{d}{c})$. The pdf and expected value follows by the usual statistical descriptions of this distribution, which can also be found in [13]. □

Figure 10 depicts how (d, c)-foldings affect the norms of unit vectors.

D Additional theory

In this section, we provide additional theory relevant to SimHash.

We present several well-known results regarding SimHash.

Proposition 2 (SimHash estimation)**.** *Let $x, y \in \mathbb{S}$, i.e., unit d-dimensional vectors. Denote $\theta = \arccos(|\cos(x, y)|)$. Let $v \in S^d$ be a unit vector drawn uniformly at random (according to the Haar measure, for example). Then,*

$$Pr[sgn(v^\top x) \neq sgn(v^\top y)] = \frac{\theta}{\pi}. \tag{5}$$

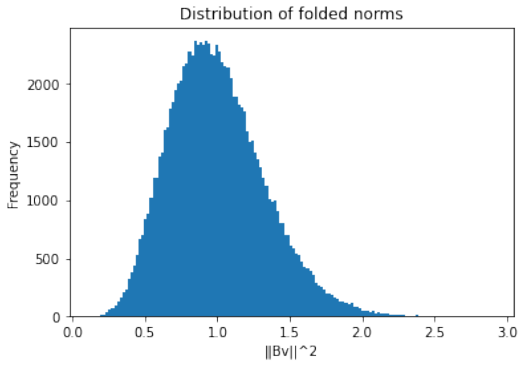


Figure 10: **Distribution of folded norms.** 100k randomly drawn unit vectors ($d = 128$) are folded down to length 16 by are usual (d, c)-folding procedure. Depicted is a binned histogram of the norms. As predicted by the statistical description of $||BX||^2$, where X is a randomly drawn unit vector, the mass is centered at 1, i.e., most norms are preserved. Empirically we observe that folded rarely exceed $\sqrt{128}16$, although the theoretical support is $[0, 8]$: this concurs with the pdf.

Proof. We reproduce the argument of [12]. We have by symmetry that $Pr[\text{sgn}(v^\top x) \neq \text{sgn}(v^\top y)] = 2Pr[v^\top x > 0, v^\top y < 0]$. The set $\mathcal{U} = \{v \in S^d : v^\top x > 0, v^\top y \leq 0\}$ corresponds to the intersection of two half-spaces whose dihedral angle (i.e., angle between the normals of both spaces) is exactly θ. Intersecting with the d-dimensional unit sphere produces gives a subspace of measure $\frac{\theta}{2\pi}$, therefore, $2Pr[v^\top x > 0, v^\top y < 0] = \frac{\theta}{\pi}$, completing the argument. □

Corollary 2. *Let v instead be a d-dimensional random Gaussian vector with iid entries $\sim \mathcal{N}(0, 1)$. Then for $x, y \in \mathbb{R}^d$,*

$$Pr[\text{sgn}(v^\top x) \neq \text{sgn}(v^\top y)] = \frac{\theta}{\pi} \tag{6}$$

Proof. Randomly drawn, normalized Gaussian vectors are well-known to be uniformly distributed on the unit sphere. □

In the setup as above, let the X be a random variable which returns 1 if x and y have differing signs when taking the standard inner product with a randomly drawn Gaussian v. Let $X_1, X_2, \ldots, X_n$ represent a sequence of independent X events. Then,

Proposition 3. $\mathbb{E}[\frac{1}{n}\sum_{i=1}^n X_i] = 1 - \frac{\theta}{\pi}$ *and* $\mathbb{V}[X] = \frac{1}{N}\frac{\theta}{\pi}(1 - \frac{\theta}{\pi})$.

Given that PGHash is equivalent to a SimHash over (d, c)-foldings of R^d, the variance reduction we observe by using multiple tables (Figure 9 is explainable by Proposition 3.