DIFFERENTIALLY PRIVATE SGD WITH SPARSE GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A large number of recent studies reveal that networks and their optimization updates contain information about potentially private training data. To protect sensitive training data, differential privacy has been adopted in deep learning to provide rigorously defined and measurable privacy. However, differentially private stochastic gradient descent (DP-SGD) requires the injection of an amount of noise that scales with the number of gradient dimensions, while neural networks typically contain millions of parameters. As a result, networks trained with DP-SGD typically have large performance drops compared to non-private training. Recent works propose to first project gradients into a lower dimensional subspace, which is found by application of the power method, and then inject noise in this subspace. Although better performance has been achieved, the use of the power method leads to a significantly increased memory footprint by storing sample gradients, and more computational cost by projection. In this work, we mitigate these disadvantages through a sparse gradient representation. Specifically, we randomly freeze a progressively increasing subset of parameters, which results in sparse gradient updates while maintaining or increasing accuracy over differentially private baselines. Our experiment shows that we can reduce up to 40% of the gradient dimension while achieve the same performance within the same training epochs. Additionally, sparsity of the gradient updates is beneficial for decreasing communication overhead when deployed in collaborative training, e.g. federated learning. When we apply our approach across various DP-SGD frameworks, we maintain accuracy while achieve up to 70% representation sparsity, which proves that our approach is a safe and effective add-on to a variety of methods. We further notice that our approach leads to improvement in accuracy in particular for large networks. Importantly, the additional computational cost of our approach is negligible, and results in reduced computation during training due to lower computational cost in power method iterations.

1 INTRODUCTION

The success of machine learning, and deep neural networks in particular, combined with ubiquitous edge computation and digital record keeping, has led to a surge in privacy sensitive learning applications. Internet-scale data promises to accelerate the development of data-driven statistical approaches, but the need for privacy constrains the amalgamation of such datasets. Private data are in fact isolated, constraining our ability to build models that learn from a large number of instances. On the other hand, the information contained in locally stored data can also be exposed through releasing the model trained on a local dataset (Fredrikson et al., 2015; Shokri et al., 2017), or even reconstructed when gradients generated during training are shared (Zhu et al., 2019; Geiping et al., 2020; Zhu & Blaschko, 2021).

To address these issues, many applications of machine learning are expected to be privacy-preserving. While differential privacy (DP) provides a rigorously defined and measurable privacy guarantee for database operations (Dwork & Roth, 2014), it also contains intriguing properties, such as robustness to post-processing and composability, which enables conveniently computing an overall privacy guarantee for several DP components. Differential privacy[1] defines privacy with respect

[1] In this work we only consider approximate differential privacy which includes the δ term.

to the difficulty of distinguishing the outputs. For a pair of neighboring databases $D, D' \in \mathcal{D}$, i.e. D can be obtained from D' by adding or removing an element.

Definition 1 *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ε, δ)-differentially private, if for any subset of outputs $S \subseteq \mathcal{R}$ it holds that:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta.$$

A common paradigm for a randomized mechanism $\mathcal{M}$ in deep learning is perturbed gradient descent:

$$\mathcal{M}(D) := f(D) + \mathcal{N}(0, S_f^2 \sigma^2 \boldsymbol{I}) \tag{1}$$

where $f : \mathcal{D} \to \mathcal{R}$ computes an aggregated gradient given a database D or D'. The isotropic Gaussian distributed noise $\xi_{DP} \sim \mathcal{N}(0, S_f^2 \sigma^2 \boldsymbol{I})$ is calibrated to f's sensitivity S_f^2, which is the maximal ℓ_2 distance $\|f(D) - f(D')\|_2$, i.e. the maximal ℓ_2 norm of gradient among all individual examples. D, D' could be batches of training data, for instance, in our experiments they are batches of image-label pairs. The factor σ is noise multiplier controlling the strength of privacy guarantee: higher σ leads to lower privacy loss. Differentially private stochastic gradient descent (DP-SGD) upper bounds the certainty of connecting data with arbitrary subset of gradient space using the privacy budget variables (ε, δ).

Bassily et al. (2014) shows that in a convex setting, DP-SGD achieves excess risk of $\tilde{O}(\sqrt{p}/n\varepsilon)$ for a model $w \in \mathbb{R}^p$ that minimizes the empirical risk $\sum_{i=1}^{n} \ell(w, d_i)$, where $d_1, d_2, ..., d_n$ are drawn from $\mathcal{D}$. While we show that in non-convex general setting, the mean square error (MSE) of perturbed gradient $\tilde{g} = g + \xi_{DP}$ is between $\Omega(p)$ and $\Omega(p^2)$ by assuming the gradients follow a Gaussian distribution:

Theorem 1 *Assuming that the gradient is drawn from $\mathcal{N}(\nabla w, \Sigma)$, centered at the true gradient ∇w and with respect to the covariance matrix Σ whose trace goes linearly up with dimension p. The MSE of perturbed gradient $\tilde{g} = g + \xi_{DP}$ can be lower bounded by:*

$$\mathbb{E}[\text{MSE}] \geq \text{Tr}[\Sigma](1 + p\sigma^2). \tag{2}$$

For the proof of this theorem, refer to Appendix A. We note that $\text{Tr}[\Sigma]$ is the trace of a p-dimensional covariance matrix. In the extreme case that $\Sigma = \lambda I$, this will scale linearly in p, while in the opposite extreme of a rank deficient covariance matrix, the trace is constant in p. Thus, in the former case

$$\mathbb{E}[\text{MSE}] \geq \lambda p + \lambda \sigma^2 p^2, \tag{3}$$

while in the latter we have

$$\mathbb{E}[\text{MSE}] \geq \lambda + \lambda \sigma^2 p, \tag{4}$$

from which we conclude that the lower bound on expected MSE is between linear and quadratic in p in practice. In terms of deep learning, as p is a large number for modern network architectures, this can lead to a significant increase in error.

The work of Abadi et al. (2016) proposed to clip the gradient of each individual example in ℓ_2 norm to a preset bound C, i.e. $\bar{g} = g \cdot \min(1, \frac{C}{\|g\|_2})$. They then apply this clipping bound to compute the variance of Gaussian distributed noise. The Gaussian noise mechanism can be expressed as:

$$\mathcal{M}(D) := f(D) + \mathcal{N}(0, C^2 \cdot \sigma^2 \boldsymbol{I}_p). \tag{5}$$

DP-SGD with gradient clipping has been empirically verified to be effective as it constraints the amount of injected noise by setting a small clipping bound. However, clipping removes the magnitude information of the gradient and therefore results in gradient estimation bias. Chen et al. (2020b) show that when the gradient distributions are near-symmetric then the clipped gradient is aligned with the true gradient and empirically demonstrate the symmetry of the gradient after several training epochs. Nevertheless, the impact of clipping bias is non-negligible in practice. Setting a small clipping bound with a deeper network will not result in better performance.

Moreover, the expected MSE of the perturbed gradient is only constrained to $O(p)$, which prevents DP-SGD approaches from being applied in truely deep networks. The so far biggest network is a CNN with Tanh proposed by Papernot et al. (2021), which reaches $\sim 66\%$ accuracy on CIFAR10 in a low privacy regime and is regarded as the state-of-the-art end-to-end network with DP-SGD. To address this curse of dimensionality, most recent works concentrate on gradient dimension reduction.

1.1 GRADIENT DIMENSION REDUCTION

Abadi et al. (2016) propose to pretrain a network on an auxiliary dataset and then transfer the feature extraction, so that a linear classifier will be replaced and trained on the private data. As CNN layers account for the majority of parameters, this results in a significant dimensionality reduction for the private setting. However, this DP-Transfer learning approach requires an auxiliary dataset with a similar distribution, and limits the capacity of the resulting network to model novel data.

Tramer & Boneh (2021) present another option. They adopt ScatterNet (Oyallon et al., 2019) to extract handcrafted features and train a relatively shallow network based on the features. Their Handcrafted CNN also mitigates the curse of dimensionality but similarly constrains the learning ability of network.

Inspired by the empirical observation that the optimization trajectory is contained in lower-dimensional subspace (Vogels et al., 2019; Gooneratne et al., 2020; Li et al.), two recent works (Zhou et al., 2021; Yu et al., 2021) intend to reduce p through Gradient Embedding Perturbation (GEP). In particular, they use an auxiliary dataset to generate a set of sample gradients, then apply the power method to find a gradient subspace and inject noise to the gradient that has been projected to the subspace. GEP also requires an auxiliary dataset, but reducing the dimensionality enables networks trained with DP-SGD to maintain reasonable performance on networks with more parameters and therefore more layers. However, because of the utilization of the power method, assuming that the number of sample gradients and the dimension of subspace are constant, the additional computational cost and memory footprint for GEP is $O(p)$ with a factor depending on the number of sample gradients and subspace dimension (which itself may scale with p). As a result, to this point GEP has mainly been applied in practice to small networks, and runs much slower than other frameworks. In our experiments GEP runs $20\times$ longer than full gradient perturbation.

In addition to the aforementioned works, McMahan et al. (2017); Yang et al. (2019) and others study how to incorporate differential privacy in collaborative training, e.g. federated learning, in the interest of protecting the privacy of participants. Federated learning is usually deployed on edge devices and local models are periodically synchronized, so communication cost becomes expensive both in time and power usage (Pathak et al., 2012). Therefore, gradient dimension reduction can have large benefits in federated learning involving power-restricted edge devices.

1.2 OUR CONTRIBUTION

In this work, we demonstrate an axis-aligned gradient dimension reduction method. In particular, we randomly zero-out a fraction of gradient during training and force the gradient to have a sparse representation. As the selected parameters are not updated in the next rounds of SGD, we name this method *random freeze*. Although simple, random freeze turns out to be effective in practice. We maintain accuracy when we adapt GEP with random freezing, while we reduce the computational cost and memory footprint induced by the power method. Applying it to various frameworks, we achieve a high representation sparsity of gradients without a loss in performance. Federated learning can take advantage of the resulting sparse representation to reduce communication costs. We further see that the random freeze strategy improves the accuracy of large networks, which we demonstrate with DP-SGD on the End-to-end CNN proposed by Papernot et al. (2021).

Finally, we also empirically study the impact on the network of injected gradient noise, and provide insights on why the network maintains good performance with sparse gradient updates.

2 APPROACH

Algorithm 1 outlines our approach. Consider a network with parameters $w \in \mathbb{R}^p$, freeze mask $m \in \{0,1\}^p$ and freeze rate r, we randomly draw rp indices and set these positions in the mask to 0 so that $\sum m = (1-r)p$. When optimizing, the network is updated with sparse gradient $g = m \odot g$. Since the indices that have been drawn do not depend on the dataset, they can be exposed or transferred in clear text and there is no additional privacy loss to the dataset caused by the random freeze strategy. We apply this strategy to optimization with SGD with or without momentum. In case of non-zero momentum, the velocity is updated with the sparse gradient as normal. That means, parameters that have been frozen can still be updated as long as their velocity

Algorithm 1: Random freeze

Input: Initialized parameters: $\mathbf{w}_0$; Loss function: ℓ; Iterations per epoch: T; Epochs E; Freeze rate: r^*; Cooling time: e^*; Clipping bound: C; Momentum: μ; Learning rate γ.

for *e = 0...E-1* **do**
  $r(e) = r^* \cdot \min(\frac{e}{e^*-1}, 1)$;
  Randomly generate a freeze mask $m \in \{0,1\}^p$ subject to $\sum m = p \cdot (1 - r(e))$;
  for *t = 0...T-1* **do**
    For each d_i in minibatch of size B, compute $g_t(d_i) = \nabla \ell(w_t, d_i)$;
    Partially zero out each gradient $g_t(d_i) = m \odot g_t(d_i)$;
    Clip each individual gradient $\bar{g}_t(d_i) = g_t(d_i) \cdot \min(1, \frac{C}{\|g_t(d_i)\|_2})$;
    Add noise $\tilde{g}_t = \frac{1}{B}(\sum_i \bar{g}_t(d_i) + m \odot \mathcal{N}(0, C^2\sigma^2 \boldsymbol{I}_p))$;
    Update $v_{t+1} = \mu \cdot v_t + g_t, w_{t+1} = w_t - \gamma v_{t+1}$;
  end
end

has not decayed to zero. When adapting GEP with random freeze, we compress the sample gradients and data gradients to dense vectors and then implement gradient embedding perturbation, so basis computation and gradient projection will be implemented in a subspace with dimension $(1 - r)p$ instead of p. Therefore, computational cost and memory footprint are accordingly reduced.

Next, we discuss the properties and implementation details of random freeze, compare different variants and provide insight into the strategy,

2.1 Gradual cooling

We find that if we initiate the training with constant freeze rate r, the network converges slowly and performs poorly when the privacy budget has been fully consumed. The reason is that in the early stages of training, the network is far from its optimal position, it is better to let all parameters stay active. So we present gradual cooling, which is inspired by the gradual warm-up of the learning rate adopted for non-privacy-preserving training (Goyal et al., 2017). Gradual cooling linearly ramps up the freeze rate from 0 to r^* within a predefined cooling time e^* and stays at r^* for the remaining training epochs, i.e. $r = r^* \cdot \min(\frac{e}{e^*-1}, 1)$.

2.2 Per-iteration randomization vs. Per-epoch randomization

Although conducting random freeze leads to negligible additional computational cost and memory footprint, we find that it is sufficient to generate one freeze mask per epoch, while re-randomizing the freeze mask at each iteration slightly decreases the performance. As we will show in Section 2.3, noise dominates throughout training. Therefore, by per-iteration randomization, parameters have been perturbed at this round could be frozen during subsequent iterations and stay biased. By contrast, per-epoch randomization lets the selected parameters update for one epoch, allowing the noise of multiple iterations to be averaged out.

Another strength of per-epoch randomization is that for one epoch there are certainly $(1 - r)p$ parameters updated, which is favorable in collaborative learning schemes as communication cost is a significant issue, and data are transmitted at the end of each epoch or several epochs. While for per-iteration randomization, the number of updated parameters for one epoch depends on the freeze rate r and iterations per epoch, resulting in higher communication overheads than per-epoch randomization.

2.3 Random freeze vs. Ranked freeze

Random freeze is an straightforward yet effective method. An intuitive variant instead would to select a crucial subset of parameters to train. We therefore consider ranking the dimensions of the gradient by their magnitude and freezing the smallest ones. For non-private SGD it has been observed that the gradients lie on low dimensional manifold (Vogels et al., 2019; Gooneratne et al.,

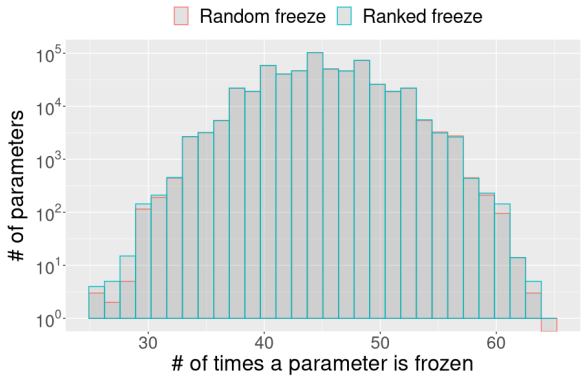


Figure 1: Histogram of the number of parameters versus the number of times a parameter is frozen.

2020; Li et al.). However for DP-SGD, to avoid extra privacy loss, we can only compute the principal components based on perturbed gradients and, as a result, subspace estimation is highly inaccurate.[2] As a result, in previous works an auxiliary dataset has been introduced as a surrogate (Zhou et al., 2021; Yu et al., 2021). Nevertheless, freezing the parameters with respect to the mean of past perturbed gradients instead of a random draw might be helpful: First, taking the mean of past perturbed gradients will average out zero-mean noise. Second, the magnitude of the gradients is a diagonal approximation to the principal components and may be indicative of a useful working subspace.

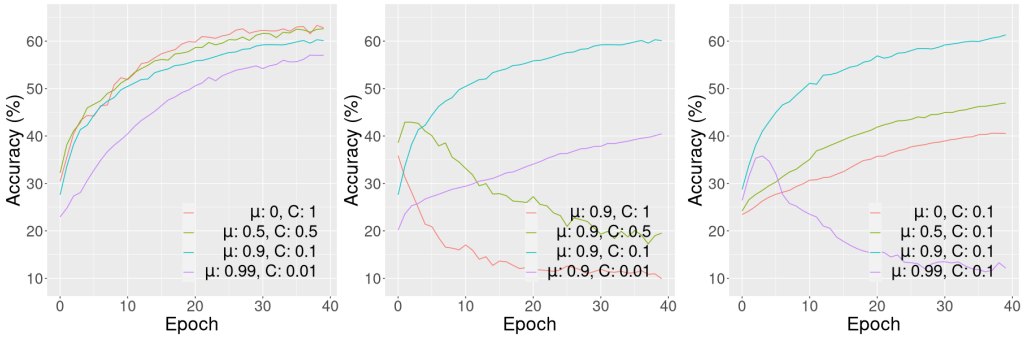
However, empirical results do not match these intuitions. We run random freeze and ranked freeze[3] on End-to-end CNN for 80 epochs, then statistically analyse the distribution of how many times a parameter is frozen. The result shows the equivalency between these two strategies, which implies even averaging the perturbed gradients over a full epoch cannot sufficiently mitigate the noise added in gradient perturbation (see Figure 1). Therefore, ranked freeze is itself inherently random as the ranking is dominated by Gaussian noise. This result also reflects that *even in a low privacy regime, noise has a significant impact throughout training*.

2.4 Inversely proportional scaling rule for adjusting the clipping bound and momentum

Previous works commonly adopt first order momentum in the optimization, because momentum can alleviate oscillation and accelerate gradient descent (Sutskever et al., 2013). As a result, it is believed to reduce the number of iterations of training and therefore achieve less privacy loss. However, for privacy-preserving training, momentum will also exaggerate the additive i.i.d. Gaussian noise by incorporating current and all historical noise. For instance, using the Pytorch (Paszke et al., 2019) implementation of SGD, the velocity update can be written as: $v_{t+1} = \mu \cdot v_t + g_{t+1}$, where v, μ, g denote perturbed velocity, momentum and perturbed gradients, respectively. Using the expression of one step noise in Equation 5 and denoting by $\hat{v}_t$ the velocity after separating the noise, we have $v_{t+1} - \hat{v}_{t+1} = (1+\mu+\mu^2+...+\mu^t) \cdot \mathcal{N}(0, C^2 \cdot \sigma^2 \boldsymbol{I}_p)$. After many iterations, the scalar approximates a geometric series, i.e. $v_{t+1} - \hat{v}_{t+1} \approx \frac{1}{1-\mu} \cdot \mathcal{N}(0, C^2 \cdot \sigma^2 \boldsymbol{I}_p)$. Pulling the clipping bound C out and forming the noise as $\frac{C}{1-\mu} \cdot \mathcal{N}(0, \sigma^2 \boldsymbol{I}_p)$, we present an inversely proportional scaling rule for adjusting C and $1-\mu$, i.e. with other hyperparameters fixed, networks trained with same value of the ratio $\frac{C}{1-\mu}$ perform similarly (see Figure 2). Our conjecture is that the inversely proportional scaling rule ensures the same amount of injected noise.

It is worth noticing that the inversely proportional scaling rule is general and amenable to no freeze and random freeze. Tuning hyperparameters in the context of privacy-perserving training has been observed to be brittle. This rule helps reduce the tuning workload.

[2]Zhou et al. (2021) argue that the reconstruction error scales with $\sqrt{p}$.
[3]For implementation details of ranked freeze, refer to Appendix B.



(a) $\frac{C}{1-\mu} = 1$ (b) $\mu = 0.9$ (c) $C = 0.1$

Figure 2: Test accuracy with respect to various clipping bound and momentum pairs. The network architecture is End-to-end CNN, and the privacy budget is $(\varepsilon = 3, \delta = 10^{-5})$. We adjust the clipping bound C and momentum μ based on their optimal values $C = 0.1, \mu = 0.9$ (Tramer & Boneh, 2021), (a) is adjusted with respect to an inversely proportional scaling rule $\frac{C}{1-\mu} = \frac{0.1}{1-0.9} = 1$, (b) has momentum fixed $\mu = 0.9$ and varies the clipping bound, (c) has clipping bound fixed to $C = 0.1$ and varies momentum. These figures demonstrate that the inversely proportional scaling rule can achieve good performance, otherwise the network performance is degraded. We also observe that no momentum $\mu = 0$ results in better performance, probably due to less clipping bias induced by a higher clipping bound.

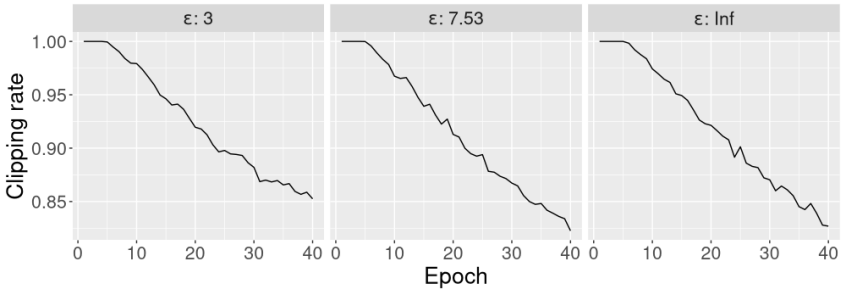


Figure 3: Clipping rate measured by the number of clipped gradients accumulated over all gradients. The network architecture is End-to-end CNN and we adopt the optimal clipping bound specified in (Tramer & Boneh, 2021). For comparison we plot ε being infinite, i.e. non-private training, where the number of gradients with norm greater than the clipping bound is counted. We show that in practice, the optimal clipping bound is relatively small. As a result most gradients have been clipped in DP-SGD.

2.5 EMPIRICAL STUDY OF RANDOM FREEZE

For non-privacy-preserving training, network pruning techniques can reduce the parameters of a network without compromising accuracy (LeCun et al., 1990; Hassibi & G.Stork, 1992; Han et al., 2015; Liu et al., 2019). Frankle & Carbin (2019) articulated the lottery ticket hypothesis and argued that networks contain subnetworks as capable as the original one. Although they reduce dimension and maintain accuracy, it is not possible to reproduce these methods with DP-SGD, because pruning requires further training or re-training while DP-SGD limits the total training epochs in consideration of restricting the amount of noise. The random freeze strategy does not prune the network. However, it provides a sparse gradient representation for optimization and obtains equal performance within the same number of training epochs as a dense gradient representation. We analyse the impact of DP-SGD on the random freeze strategy in two aspects in the sequel.

First, DP-SGD makes the network more redundant as it simplifies the task. There exist techniques leveraging injected noise to overcome overfitting and improve test accuracy or approximate global

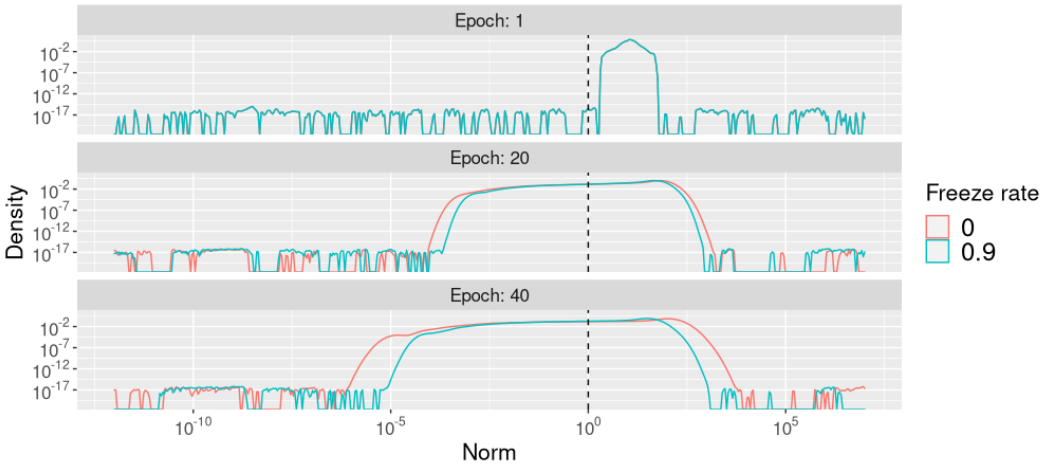


Figure 4: Gradient norm distribution of DP-SGD with or without the random freeze strategy. The vertical dashed line indicates the clipping bound. With the random freeze strategy, in later epochs the variance of the norm magnitude decreases. A lower number of high-magnitude gradient norms implies less clipping bias, while the decrease in low magnitude gradient norms implies a higher signal-to-noise ratio of the perturbed gradients. The two plots overlap in the subfigure corresponding to the first epoch as the freeze rate is 0 and the networks are initialized equally. The freeze rate at the 20th epoch is 0.45 and reaches 0.9 at the 40th epoch. Note that both axes are in log scale.

optimization in a large search space, e.g. Stochastic Gradient Langevin Dynamics (Welling & Teh, 2011) and Simulated Annealing. Both techniques require a calibrated amount of noise, which is in practice several magnitudes lower than that required by DP-SGD. As we have shown in Section 2.3, in low privacy regime noise will dominate through the whole training phase. Excessive noise will smooth the boundaries between classes. Moreover, differentially private learning clips large gradients and the clipping rate can be regularly more than 80% in practice (see Figure 3). Clipping removes the magnitude information of gradients, and as a result boundaries are further smoothed. Consequently, DP-SGD leads to a significant drop in accuracy, and a DP network is more redundant than an equivalent non-private network. Searching for an optimal network in a space restricted by the random freeze strategy should still be effective.

Second, compared to no freezing, random freeze leads to less clipping bias and gradient distortion, as shown in Figure 4. We adopt the same clipping bound for random freeze and no freeze. As fewer dimensions contribute to the norm computation, random freeze reduces the clipping probability and therefore alleviates clipping bias (Zhang et al., 2020). We also observe that the norm of sparse gradients are not equally scaled down, weak gradients can spontaneously become larger during training, which mitigates the distortion of gradients due to perturbation. Note that the dependence between the freezing rate and the expected norm of injected Gaussian noise is proportional to $\sqrt{1-r}$.

3 Experimental Results

According to Definition 1, we have that each iteration of training is (ε, δ)-differentially private with respect to a batch of training data. Abadi et al. (2016) proposed to account for shuffling and partitioning the dataset into batches, which implies that each iteration is $(q\varepsilon, q\delta)$-differentially private with respect to the full dataset according to the privacy amplification theorem (Beimel et al., 2010; Kasiviswanathan et al., 2011), where $q = B/N$, B is the batchsize and N is the size of dataset. This assumption has been widely accepted for differentially private learning. To track the cumulative privacy loss over multiple training epochs, we adopt Rényi differential privacy (Mironov, 2017), which is more operationally convenient and quantitatively accurate compared with the original definition of differential privacy. Rényi differential privacy can also be converted to the conventional (ε, δ)-differential privacy expression.

			Test Accuracy		
Approaches	ε-DP	# of parameters	Baseline	Random freeze	Freeze rate r^*
End-to-end CNN	7.53	550K	66.9 ± 0.4	66.7 ± 0.3	0.7
	3.0		60.4 ± 0.2	61.1 ± 0.2	0.7
Handcrafted CNN	3.0	187K	69.4 ± 0.2	69.4 ± 0.2	0.6
GEP	8.0	268K	73.5 ± 0.4	73.4 ± 0.3	0.4
DP-Transfer Learning	2.0	41K	92.6 ± 0.0	92.7 ± 0.0	0.7

Table 1: Test accuracy of SOTA works before and after adopting random freeze. We maintain the accuracy with high freeze rate. Communication overhead, computational cost and memory footprint of the power method are accordingly reduced as implied by total density in Table 2.

Total density			
End-to-end CNN	Handcrafted CNN	GEP	DP-Transfer Learning
0.65	0.7	0.8	0.65

Table 2: Total representation density of random freeze. This table is aligned with Table 1.

Our experiments are implemented in the Pytorch framework (Paszke et al., 2019). To compute the gradients of an individual example in a minibatch, which is required for gradient clipping, we use the BackPACK package (Dangel et al., 2020). The privacy loss of multiple iterations has been tracked with Opacus. We use the benchmark CIFAR10 (Krizhevsky, 2012), which is to date standard in benchmarking DP learning. Our code is available for download from `http://anonymized.for.review/`.

To validate the reliability of random freeze, we conduct experiments on several SOTA networks cross different frameworks, including End-to-end CNN (Papernot et al., 2021); Handcrafted CNN (Tramer & Boneh, 2021) which incorporates ScatterNet (Oyallon et al., 2019) as feature extractor; Gradient Embedding Perturbation (GEP) proposed by Yu et al. (2021) which injects noise in a low dimensional manifold; DP-Transfer Learning which adopts a pretrained network and replaces the linear classifier layer, in particular we use SIMCLR v2 (Chen et al., 2020a) pretrained on unlabeled ImageNet (Deng et al., 2009), which has been benchmarked by Tramer & Boneh (2021).

For a fair comparison, we run every experiment 5 times then compute the average of best accuracy and the standard error. It is worth noticing when adapting the SOTA works with random freeze, we do not tune the hyperparameters[4], instead we adopt the optimal hyperparameters provided in the respective works. Random freeze is applied as following: the optimal number of epochs from the previous work is set as e^*, so we linearly ramp up the freeze rate r from $r = 0$ at epoch $e = 0$ to $r = r^*$ at epoch $e = e^*$, i.e. $r = r^* \cdot \frac{e}{e^*-1}$. Although naive, such experiments allow us to demonstrate that random freeze is a safe add-on in a variety of methods. We define total density as the total amount of non-zero gradients by random freeze over the total amount of gradients by the original dense representation. Total density reflects two advantages of random freeze: First, for GEP the overall computational cost and memory footprint are correspondingly reduced; Second, for federated learning the total amount of data that needs to be transferred is correspondingly reduced. We summarize the performance results in Table 1, and the corresponding total density by random freeze in Table 2. We document the hyperparameters in Appendix C.

We find that when the network is large, for instance, End-to-end CNN has the most parameters among all frameworks, random freeze can also improve the accuracy. To demonstrate this, we further tune the End-to-end CNN. We still adopt the best hyperparameters from the previous work, then first adjust the clipping bound and momentum from $(C = 0.1,\ \mu = 0.9)$ to $(C = 1,\ \mu = 0)$ with respect to the inversely proportional scaling rule proposed in section 2.4, which leads to better

[4]There exist too ways of determining the injected noise scale per iteration: 1. Set the privacy budget and noise multiplier then stop the training when the privacy budget is fully consumed; 2. Set the privacy budget and iterations of training then compute the noise multiplier so that the privacy budget will be fully consumed at the last iteration. In this work we refer to the second method when talking about hyperparameters.

		Test Accuracy		
Approaches	ε-DP	Baseline	Our adjusted baseline	Random freeze
End-to-end CNN	7.53	66.9 ± 0.4	69.7 ± 0.1	$\mathbf{70.2 \pm 0.1}$
	3.0	60.4 ± 0.2	63.1 ± 0.2	$\mathbf{64.5 \pm 0.3}$

Table 3: Test accuracy of End-to-end CNN adjusted with respect to inversely proportional scaling rule and trained with random freeze. Our adjusted baseline performs better than original work while with random freeze we further improve the accuracy. Altogether we obtain significantly better utility.

accuracy. Secondly, we apply random freeze. Similar to the previous experiment we also linearly ramp up the freeze rate to r^* but then we extend the training for 20 additional epochs at freeze rate r^*, this help us to improve the performance further. The result is summarized in Table 3. Note that, after extending the training epochs, the noise multiplier σ has been increased accordingly to ensure the same privacy budget will be consumed. As a result, more epochs without random freeze will decrease the performance. Moreover, we achieve lower total density with higher freeze rate. The hyperparameters and total density as well as accuracy of each classes are documented in Appendix D.

4 Discussion and Conclusions

In this work we demonstrate an axis-aligned random gradient dimension reduction method. Although simple to implement, the random freeze strategy can be safely (without further tuning) applied across different frameworks and architectures. Even in conjunction with other gradient space reduction methods performance is maintained. In addition to these advantages, because of the sparse representation of the gradient update, random freeze can reduce the computational cost and memory footprint of the power method in GEP. Also, it is able to reduce the total amount of transferred data in federated learning or other collaborative learning frameworks. Moreover, we significantly improve the performance of SOTA End-to-end CNN using the random freeze strategy and inversely proportional scaling rule that we have proposed. We note that the computational cost of the random freeze procedure is negligible. Therefore we believe that random freeze can be incorporated as a general method in DP-SGD. Our inversely proportional scaling rule helps to reduce the tuning workload of DP-SGD. Additionally, we provide many insights on random freeze and DP-SGD.

It is interested to observe that in providing much less gradient information we can still maintain performance in the same number of optimization iterations. When the network architecture is large, we can even improve the performance with this gradient dimension reduction method. We expect random freeze to exhibit strong improvements as DP-SGD is adapted to larger and deeper networks in the future. We also hope this work can shed light on gradient dimension reduction in DP-SGD and motivate further research in this direction.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pp. 464–473, 2014.

Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In *Proceedings of the 7th International Conference on Theory of Cryptography*, pp. 437–454, 2010.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems 33*, 2020a.

Xiangyi Chen, Steven Z. Wu, and Mingyi Hong. Understanding gradient clipping in private SGD: A geometric perspective. In *NeurIPS*, 2020b.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *International Conference on Learning Representations*, 2020.

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

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9:211–407, 2014.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, pp. 16937–16947, 2020.

Mary Gooneratne, Khe Chai Sim, Petr Zadrazil, Andreas Kabel, Françoise Beaufays, and Giovanni Motta. Low-rank gradient approximation for memory-efficient on-device training of deep neural network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3017–3021, 2020.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. 06 2017.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. NIPS'15, pp. 1135–1143, 2015.

Babak Hassibi and David G.Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Adv Neural Inform Proc Syst*, 1992.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40:793–826, 2011.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2012.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, 1990.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. *Hessian based analysis of SGD for Deep Nets: Dynamics and Generalization*, pp. 190–198.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

Arak Mathai. Storage capacity of a dam with gamma type inputs. *Annals of the Institute of Statistical Mathematics*, pp. 591–597, 1982.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Ilya Mironov. Rényi differential privacy. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017.

Panagis Moschopoulos. The distribution of the sum of independent gamma random variables. *Annals of the Institute of Statistical Mathematics*, pp. 541–544, 1985.

P.G. Moschopoulos and W.B. Canada. The distribution function of a linear combination of chi-squares. *Computers & Mathematics with Applications*, pp. 383–386, 1984.

Opacus. Opacus PyTorch library. Available from opacus.ai.

Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, and Eugene Belilovsky. Scattering networks for hybrid representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 9312–9321, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app? fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pp. 29–42, 2012.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1139–1147, 2013.

Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2021.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. *PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization*. 2019.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. ICML'11, pp. 681–688, 2011.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations*, 2021.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

Yingxue Zhou, Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private SGD with gradient subspace identification. In *International Conference on Learning Representations*, 2021.

Junyi Zhu and Matthew B. Blaschko. R-GAP: Recursive gradient attack on privacy. In *International Conference on Learning Representations*, 2021.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, 2019.

A MEAN SQUARE ERROR OF GRADIENT IN DP-SGD

Assuming an oracle can tell us the true gradient ∇w given a network with parameters $w \in \mathbb{R}^p$, while the individual gradient g can be seen as drawn from the Gaussian distribution centered at ∇w with respect to the covariance matrix Σ, i.e. $g \sim \mathcal{N}(\nabla w, \Sigma)$. Denote ξ_g as gradient deviation, i.e. $\xi_g = g - \nabla w$, the perturbed gradient $\tilde{g}$ in DP-SGD can be expressed as:

$$\tilde{g} = \nabla w + \xi_g + \xi_{DP} \tag{6}$$

and the expected mean square error of perturbed gradient is:

$$\mathbb{E}[\text{MSE}] = \mathbb{E}[\xi_g^\intercal \xi_g + 2\xi_g^\intercal \xi_{DP} + \xi_{DP}^\intercal \xi_{DP}] \tag{7}$$

Since ξ_{DP} and ξ_g are independent and plugging the definition of injected noise. i.e. $\xi_{DP} \sim \mathcal{N}(0, \max(g^\intercal g)\sigma^2 \boldsymbol{I}_p)$, into Equation 7, we have:

$$\begin{aligned}
\mathbb{E}[\text{MSE}] &= \mathbb{E}[\xi_g^\intercal \xi_g] + \mathbb{E}[\xi_{DP}^\intercal \xi_{DP}] && (8)\\
&= \text{Tr}[\Sigma] + p\sigma^2 \mathbb{E}[\max(g^\intercal g)] && (9)\\
&= \text{Tr}[\Sigma] + p\sigma^2 \mathbb{E}[\max(\xi_g^\intercal \xi_g + 2\nabla w^\intercal \xi_g + \nabla w^\intercal \nabla w)] && (10)\\
&\geq \text{Tr}[\Sigma] + p\sigma^2 \mathbb{E}[\max(\xi_g^\intercal \xi_g)] && (11)
\end{aligned}$$

where equality can be achieved when the network converges to stationary point, i.e. $\nabla w \to \mathbf{0}$.

Applying a spectral decomposition to Σ, we obtain $Q^\intercal \Lambda Q = \Sigma$, where Q is an orthonormal basis and Λ is diagonal matrix of eigenvalues of Σ. Denote by z a random vector drawn i.i.d. from a normal distribution, we can further derive:

$$\xi_g^\intercal \xi_g \sim z^\intercal \Sigma z = z^\intercal Q^\intercal \Lambda Q z = z^\intercal \Lambda z \tag{12}$$

As random variable with respect to the same distribution leads to the same expectation, plugging Equation 12 into Equation 11, we have:

$$\begin{aligned}
\mathbb{E}[\text{MSE}] &\geq \text{Tr}[\Sigma] + p\sigma^2 \mathbb{E}[\max(z^\intercal \Lambda z)] && (13)\\
&= \text{Tr}[\Sigma] \left(1 + p\sigma^2 \mathbb{E}\left[\max\left(\frac{1}{\text{Tr}[\Sigma]} z^\intercal \Lambda z\right)\right]\right) && (14)\\
&= \text{Tr}[\Sigma] \left(1 + p\sigma^2 \mathbb{E}\left[\max\left(\frac{1}{\text{Tr}[\Lambda]} z^\intercal \Lambda z\right)\right]\right) && (15)
\end{aligned}$$

Since $z^\intercal \Lambda z = \lambda_1 z_1^2 + \lambda_2 z_2^2 + \cdots + \lambda_d z_p^2;\ z_i \sim \mathcal{N}(0, 1)$, we have:

$$\frac{1}{\text{Tr}[\Lambda]} z^\intercal \Lambda z \sim \frac{1}{\text{Tr}[\Lambda]} \sum_i \Gamma\left(\frac{1}{2}, \frac{1}{2\lambda_i}\right). \tag{16}$$

So far either the exact PDF of a Gamma distribution summation can only be represented with (another) series of gamma distributions (Moschopoulos, 1985; Mathai, 1982), or its exact CDF has a series representation (Moschopoulos & Canada, 1984). For the convenience of further derivation of the expected maximum, we use the Welch-Satterthwaite approximation which provides a certain Gamma distribution:

$$\begin{aligned}
\frac{1}{\text{Tr}[\Lambda]} \sum_i \Gamma\left(\frac{1}{2}, \frac{1}{2\lambda_i}\right) &\approx \frac{1}{\text{Tr}[\Lambda]} \Gamma\left(\frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}, \frac{\text{Tr}\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}\right) && (17)\\
&= \Gamma\left(\frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}, \frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}\right) && (18)
\end{aligned}$$

Denote $M_n = \max(x_1, x_2, ..., x_n)$, where x_i is drawn from $\Gamma\left(\frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}, \frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}\{\Lambda^2\}}\right)$. While from the Cauchy-Schwarz inequality, we have:

$$\begin{aligned}
|\langle \boldsymbol{I}_p, \text{diag}\{\Lambda\}\rangle|^2 &\leq \langle \boldsymbol{I}_p, \boldsymbol{I}_p\rangle \cdot \langle \text{diag}\{\Lambda\}, \text{diag}\{\Lambda\}\rangle && (19)\\
\text{Tr}^2[\Lambda] &\leq p \cdot \text{Tr}[\Lambda^2] && (20)\\
\frac{\text{Tr}^2\{\Lambda\}}{2\,\text{Tr}[\Lambda^2]} &\leq \frac{p}{2} && (21)
\end{aligned}$$

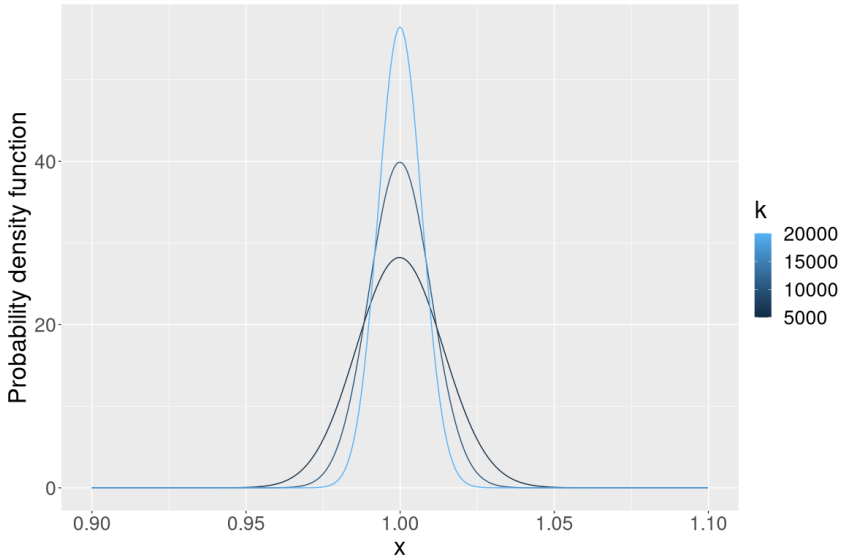


Figure 5: A series of Gamma distributions with the form $\Gamma(k, k)$.

Denote k as $\frac{\mathrm{Tr}^2\{\Lambda\}}{2\,\mathrm{Tr}\{\Lambda^2\}}$, Figure 5 illustrates the PDF of Gamma distribution with parameters $\Gamma(k, k)$.

As k becomes large, the distribution becomes more concentrated, and we have the following inequality:

$$
\begin{aligned}
M_n^{(1)} &= \max(x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}),\ x_i^{(1)} \sim \Gamma(\frac{\mathrm{Tr}^2\{\Lambda\}}{2\,\mathrm{Tr}\{\Lambda^2\}}, \frac{\mathrm{Tr}^2\{\Lambda\}}{2\,\mathrm{Tr}\{\Lambda^2\}}) \\
M_n^{(2)} &= \max(x_1^{(2)}, x_2^{(2)}, ..., x_n^{(2)}),\ x_i^{(2)} \sim \Gamma(\frac{p}{2}, \frac{p}{2}) \\
&\mathbb{E}[M_n^{(1)}] \geq \mathbb{E}[M_n^{(2)}] \geq \mathbb{E}[\sum_i x_i^{(2)}] = 1
\end{aligned} \tag{22}
$$

Substituting inequality 22 back into Equation 15, we have:

$$
\mathbb{E}[\mathrm{MSE}] \geq \mathrm{Tr}[\Sigma](1 + p\sigma^2). \tag{23}
$$

B Ranked freeze

Algorithm 2 describes ranked freeze.

C Hyperparameters for Table 1

The δ term is 10^{-5} for all settings. Note that hyperparameters ε, σ, Batchsize, Epoch are redundant, one can be derived from the rest. Hyperparameters are listed separately for each framework, see Table 4, 6, 5, 7. There is a minor difference for hyperparameters adopted from Tramer & Boneh (2021). For instance, they set batchsize=1024 for End-to-end CNN, which leads to non-exact batch partitions. We set batchsize=1000 to avoid that and achieve equally good or slightly better baseline performance.

D Improving the accuracy of End-to-end CNN

Table 8 includes the hyperparameters and total density of ajusted End-to-end CNN with random freeze. Figure 6 shows the accuracy for individual classes.

Algorithm 2: Ranked freeze

Input: Initialized parameters: $\mathbf{w}_0$; Loss function: ℓ; Iterations per epoch: T; Epochs E; Freeze rate: r^*; Cooling time: e^*; Clipping bound: C; Momentum: μ; Learning rate γ.

for *e = 0...E-1* **do**
- $\hat{g}^e = \{0\}^p$;
- $r(e) = r^* \cdot \min(\frac{e}{e^*-1}, 1)$;
- **if** *e is 0* **then**
  - $m = \{1\}^p$
- **else**
  - Sort the [1,...,p] with respect to corresponding aggregated gradient of the last epoch $\hat{g}^{e-1}$ in ascending order then set the first $p \cdot r(e)$ positions in mask m to 0 and the rest to 1;
- **end**
- **for** *t = 0...T-1* **do**
  - For each d_i in minibatch of size B, compute $g_t(d_i) = \nabla \ell(w_t, d_i)$;
  - Partially zero out each gradient $g_t(d_i) = m \odot g_t(d_i)$;
  - Clip each individual gradient $\bar{g}_t(d_i) = g_t(d_i) \cdot \min(1, \frac{C}{\|g_t(d_i)\|_2})$;
  - Add noise $\tilde{g}_t = \frac{1}{B}(\sum_i \bar{g}_t(d_i) + m \odot \mathcal{N}(0, C^2\sigma^2 \boldsymbol{I}_p))$;
  - Update $v_{t+1} = \mu \cdot v_t + g_t, w_{t+1} = w_t - \gamma v_{t+1}, \hat{g}^e = \hat{g}^e + \tilde{g}_t$;
- **end**

end

Table 4: Hyperparameters of End-to-end CNN adopted from Tramer & Boneh (2021)

End-to-end CNN

ε	σ	lr	Batchsize	Epoch	Momentum	Clip
3	1.54	1	1000	40	0.9	0.1
7.53	1.10	1	1000	80	0.9	0.1

Table 5: Hyperparameters of DP-Transfer Learning adopted from Tramer & Boneh (2021)

DP-Transfer Learning

ε	σ	lr	Batchsize	Epoch	Momentum	Clip	Source model	Aux. dataset
2	2.30	4	1000	50	0.9	0.1	SIMCLR v2	ImageNet

Table 6: Hyperparameters of Handcrafted CNN adopted from Tramer & Boneh (2021)

Handcrafted CNN

ε	σ	lr	Batchsize	Epoch	Momentum	Clip	Input norm	BN norm	Architecture
3	5.65	4	8000	80	0.9	0.1	BN	8	ScatterNet + CNN

Table 7: Hyperparameters of GEP adopted from Yu et al. (2021)

Gradient Embedding Perturbation

ε	lr	Batchsize	Epoch	Momentum	Clip0	Clip1	Subspace	Sample gradients	Aux. dataset
8	0.1	1000	200	0.9	5	2	1000	2000	ImageNet

Table 8: Hyperparameters for adjusted End-to-end CNN with random freeze

Adjusted End-to-end CNN with random freeze

ε	σ	lr	Batchsize	Epoch	Momentum	Clip	Freeze rate	Cooling time	Total density
3	1.81	1	1000	60	0	1	0.9	40	0.6
7.53	1.18	1	1000	100	0	1	0.9	80	0.58

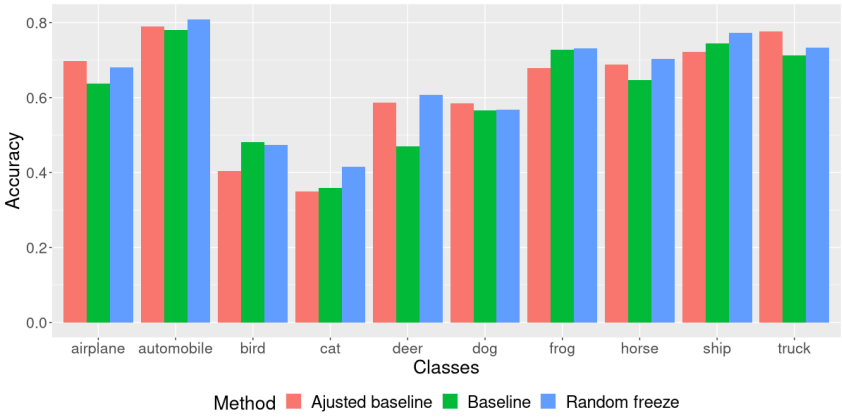


Figure 6: Accuracy of classes.