All Rivers Run to the Sea: Private Learning with Asymmetric Flows

Supplementary Material

8. Proofs of Theorem 1 and Theorem 2

In this appendix, we provide our proofs. We start with the proof of Theorem 1 in Appendix 8.1. Then, we provide the proof of Theorem 2 in Appendix 8.2.

8.1. Proof of Theorem 1

We now prove Theorem 1 showing the feasibility of designing a low-dimensional layer.

Theorem 1. *For a convolution layer with weight* $\mathbf{W} \in \mathbb{R}^{n\times c\times k\times k}$ *with an input* $\mathbf{X}$ *with rank* r *and output* $\mathbf{Y}$ *with rank* q*, there exists an optimal* $\mathbf{W}^{(1)} \in \mathbb{R}^{q\times c\times k\times k}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{n\times q\times 1\times 1}$ *in the low-dimensional layer such that the output of this layer denoted as* $\mathbf{Y}^{'}$ *satisfies*

$$\left\|\mathbf{Y} - \mathbf{Y}^{'}\right\| = 0. \tag{8}$$

Proof. Given an input tensor $\mathbf{X} \in \mathbb{R}^{c\times h\times w}$ with rank r, $\mathbf{W} \in \mathbb{R}^{n\times c\times k\times k}$, and output $\mathbf{Y} \in \mathbb{R}^{n\times h\times w}$ with rank q, we can write the convolution $\mathbf{Y} = \mathbf{W} \circledast \mathbf{X}$ in a matrix form as

$$Y = W\cdot X, \tag{9}$$

where $W \in \mathbb{R}^{n\times ck^2}$, $X \in \mathbb{R}^{ck^2\times hw}$ [11, 41, 56]. Similarly for the low-dimensional convolution layer, we can write its output $\mathbf{Y}^{'}$ as follows

$$Y^{'} = W^{(2)}\cdot W^{(1)}\cdot X,$$

where $W^{(2)} \in \mathbb{R}^{n\times q}$ and $W^{(1)} \in \mathbb{R}^{q\times ck^2}$.

According to Theorem 2 in [41], the output's rank is decided by the rank of X. Therefore, we can decompose X using SVD as

$$X = U\cdot V,$$

where $U \in \mathbb{R}^{ck^2\times q}$ is a matrix with orthogonal columns, and $V \in \mathbb{R}^{q\times hw}$ is a matrix with orthogonal rows. With the low-rank decomposition, we can express the difference between the outputs Y and $Y^{'}$ as

$$\begin{aligned} Y - Y^{'} &= W\cdot U\cdot V - W^{(2)}\cdot W^{(1)}\cdot U\cdot V \\ &= (W\cdot U - W^{(2)}\cdot W^{(1)}\cdot U)\cdot V. \end{aligned}$$

Let $Z = W\cdot U - W^{(2)}\cdot W^{(1)}\cdot U$, owing to the low-rank of the matrix U, the original kernel matrix W is reduced to a low-dimensional form. Therefore, we can express the kernel matrix W as follows

$$W = W^U + R, \tag{10}$$

where $W^U = W\times UU^*$ is a low-rank matrix obtained based on the principal components of U, while R is a residual matrix based on the principal components that are orthogonal to U. That is, $R\cdot U = 0$.
Then the difference $Z = W\cdot U - W^{(2)}\cdot W^{(1)}\cdot U$ can be written as follows

$$\begin{aligned} Z &= \left[(W^U + R) - W^{(2)}\cdot W^{(1)}\right]\cdot U \\ &= (W^U - W^{(2)}\cdot W^{(1)})\cdot U. \end{aligned}$$

Since $\texttt{Rank}\,(W^U) \le q$, then there is a solution such that

$$\min_{W^1,W^2}\left\|W^U - W^{(2)}\cdot W^{(1)}\right\| = 0,$$

where the pair $W^{(1)}, W^{(2)}$ is one of the rank-q decompositions of W^U. Hence, $\min_{W^1,W^2}\left\|\mathbf{Y} - \mathbf{Y}^{'}\right\| = 0$. □

8.2. Proof of Theorem 2

Next, we provide the proof of Theorem 2.

Theorem 2. `Delta` *ensures that the perturbed residuals and operations in the public environment satisfy* (ϵ,δ)*-DP given noise* $\mathbf{N}\sim\mathcal{N}(0, 2C^2\cdot\log(2/\delta')/\epsilon'\cdot\mathbf{I})$ *given sampling probability* p*, and* $\epsilon = \log(1 + p(e^{\epsilon'} - 1)), \delta = p\delta'$.

Proof. The proof relies on Theorem 6 in [28], the subsampling theorem in [9] (Theorem 9) (replicated in Theorem 3 and 4), and the post-processing rule of DP.

First, we show that the mechanism is (ϵ,δ)-DP if $\sigma^2 = 2C^2\cdot\log(2/\delta')/\epsilon'$. Since the mechanism of obtaining IR_{noisy} is a Gaussian mechanism, and the variance is $2C^2\cdot\log(2/\delta')/\epsilon'$, then it follows that the mechanism ensures (ϵ,δ)-DP by Theorem 3 and Theorem 4.

The residual model $\mathcal{M}_{\text{res}}$ in the public environment and outputs after $\mathcal{M}_{\text{res}}$ are the post-processing of IR_{noisy}. Since the post-processing does not affect the DP budget [17], any operation in the public environment has the same privacy budget as the Gaussian mechanism. □

Theorem 3. *(Theorem 6 in [28]) Given noise with* $\sigma^2 = 2C^2\cdot\log(2/\delta)/\epsilon$*, the Gaussian mechanism is* (ϵ,δ)*-DP.*

Theorem 4. *(Theorem 9 [9]) Given a randomized mechanism* $\mathcal{M}'$ *with privacy parameter* (ϵ',δ')*, and* $\mathcal{M}$ *with sampling probability* p*, for any* $\epsilon' > 0, \delta' > 0$*, we have* $\epsilon = \log(1 + p(e^{\epsilon'} - 1))$*, and* $\delta = p\delta'$.

9. Ablation Study

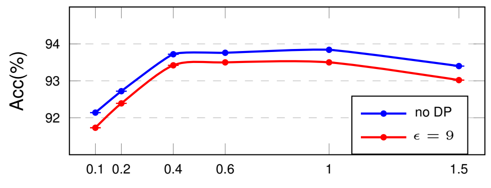
In this section, we conduct three important ablation studies. The first one explores merging logits with scaling factor, whereas the second investigates more perturbation effects on the overall model performance and finally the third one investigates the effects of the binary quantization.

9.1. Logits Merging with Scaling

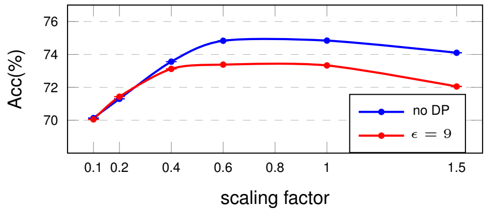
In the main paper, we directly add the logits from $\mathcal{M}_{\text{main}}$ and $\mathcal{M}_{\text{res}}$ to obtain the final prediction (See Figure 1b). In this appendix, we explore a different way of merging logits. Specifically, given logits vector from $\mathcal{M}_{\text{main}}$ and $\mathcal{M}_{\text{res}}$: $\boldsymbol{z}_{\text{main}}$, $\boldsymbol{z}_{\text{res}}$, we add a scaling coefficient α during merging as

$$\boldsymbol{z}_{\text{tot}} = \boldsymbol{z}_{\text{main}} + \alpha \cdot \boldsymbol{z}_{\text{res}}.$$

The scaling factor controls the weight of $\mathcal{M}_{\text{res}}$'s prediction. Since $\mathcal{M}_{\text{res}}$ only contains residual information, its prediction might conflict with $\mathcal{M}_{\text{main}}$ when residuals contain little information. With the scaling factor, potential conflicts between $\mathcal{M}_{\text{main}}$ and $\mathcal{M}_{\text{res}}$ can be mitigated.



(a) ResNet-18/CIFAR-10



(b) ResNet-18/CIFAR-100

Figure 8. Ablation study on merging logits with scaling. Logits merging with a small α limits the useful information from the residual model, incurring accuracy drops. Merging with a large $\alpha (> 1)$ also incurs accuracy drop as $\mathcal{M}_{\text{res}}$ can overshadow $\mathcal{M}_{\text{main}}$ in the final prediction.

Figure 8 shows the accuracy versus α for ResNet-18 on CIFAR-10/100. We next make the following observations.

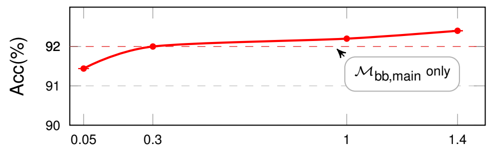
- With small α, there is a noticeable accuracy drop. The reason is that small α reduces the weight of $\mathcal{M}_{\text{res}}$'s logits, limiting information from the residual path. As a result, $\mathcal{M}_{\text{res}}$ barely provides performance improvements.
- As α increases, $\mathcal{M}_{\text{res}}$'s prediction weighs more, thereby boosting the performance of the overall model. We can also observe that there is a large adjustment space for α, which leads to optimal performance.
- With further large α $(\alpha > 1)$, $\mathcal{M}_{\text{res}}$ becomes more and more dominant and dominates the main model's prediction. The overall performance again decreases.

Therefore, the weight of $\mathcal{M}_{\text{res}}$'s logits affects the overall model performance. `Delta` in the main paper assigns equal weights for $\mathcal{M}_{\text{main}}$ and $\mathcal{M}_{\text{res}}$ $(\alpha = 1)$, which strikes an optimal balance between predictions from those two models.

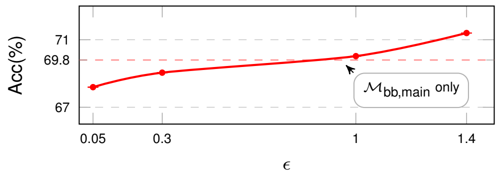
9.2. More Effects of Perturbation

In this ablation study, we investigate the potential adverse effects of the perturbed residuals on overall performance. As we add very large noise on residuals, information in residuals is significantly perturbed. As a result, the residual model $\mathcal{M}_{\text{res}}$ can cause conflicts with the main model, rather than provide additional beneficial information for the final prediction.

Figure 9 shows the accuracy of ResNet-18 on CIFAR-10/100 with small privacy budgets. With small ϵ, the noise for perturbation is very large, thereby making the residual model $\mathcal{M}_{\text{res}}$ unable to extract useful information from residuals. And it further affects the overall model accuracy. In particular, the final model accuracy can be even lower than without $\mathcal{M}_{\text{res}}$ (red dashed line in Figure 9). Note that this ablation study mainly aims to investigate more effects of perturbation under very small ϵ, but such strong privacy constraints are usually not considered in real scenarios. Furthermore, even in this case, with very strict privacy constraints, users can train $\mathcal{M}_{\text{bb}}$ and $\mathcal{M}_{\text{main}}$ only. Owing to the effective asymmetric decomposition, $\mathcal{M}_{\text{bb}}$ and $\mathcal{M}_{\text{main}}$ still give reasonable accuracy without incurring prohibitive costs in the private environment.



(a) ResNet-18/CIFAR-10



(b) ResNet-18/CIFAR-100

Figure 9. Effects of large perturbation on overall performance. $\mathcal{M}_{\text{res}}$ under strict privacy constraints can result in an overall accuracy even lower than without $\mathcal{M}_{\text{res}}$. In this case, users can train $\mathcal{M}_{\text{bb}}$ and $\mathcal{M}_{\text{main}}$ only, which achieves reasonable accuracy owing to the asymmetric decomposition.

9.3. Effects of Binary Quantization

This ablation study analyzes how the binary quantization affects the final model's performance. As elaborated in Section 4.2, the binary quantization reduces the communication cost when sending residuals to the public environment. We train ResNet-18 on CIFAR-10 and CIFAR-100 with and without binary quantization, and report their results in Table

7. The results show that binary quantization does not significantly affect the accuracy of the final model under different privacy budgets.

10. Experiment for Asymmetric Structure

In this appendix, we provide the experimental details for the asymmetric structure of the IRs in Section 3.
We train ResNet-18 with ImageNet and the hyperparameters listed in Table 8. When the training is complete, we analyze the asymmetric structures in the validation dataset.
Specifically, we extract the intermediate features after the first convolutional layer in ResNet-18, then use SVD and DCT to analyze the channel and spatial correlation as in Section 3. For DCT, since the feature size after the convolutional layer is 56×56, we use 14×14 block-wise DCT.
We compute the relative error $\frac{\|\mathbf{X}-\mathbf{X}_{\text{lr}}\|}{\|\mathbf{X}\|}$ and $\frac{\|\mathbf{X}-\mathbf{X}_{\text{lf}}\|}{\|\mathbf{X}\|}$ for each input, and average the ratio across the entire validation datasets. By varying the number of principal channels in $\mathbf{X}_{\text{lr}}$, r, and the number of low-frequency components in X_{lf}, t'^2, we obtain the results in Figure 2.

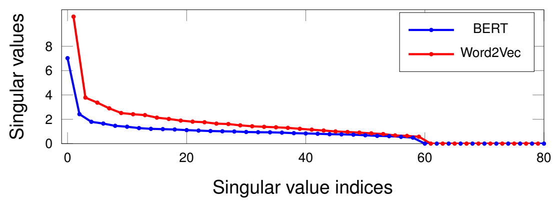
10.1. Asymmetric Structure in Language Models

The asymmetric structure of the intermediate representations is not only observed in computer vision models but also in language models. In this appendix, we show the asymmetric structure of word embedding vectors in language models.

We use Word2Vec [37] and the word embedding layer in BERT [16] to generate embedding vectors. Figure 10b shows an example text with 80 words. We feed the text to Word2Vec and BERT word embedding layer, obtaining embedding vectors. We group the embedding vectors as a matrix with each one stored in one row. With the embedding matrix obtained from Word2Vec and BERT, we respectively apply SVD to the matrix and compute their singular values, as shown in Figure 10a. We can easily observe that the decay of the singular values follows an exponential manner for both Word2Vec and the BERT embedding layer, indicating high correlations among the embedding vectors. With the principal vector after SVD, we further use the first 16 principal vectors (1/5 of the total vectors) from Word2Vec embedding and reconstruct an approximated matrix, where each row approximates the original embedding vector. Then, we reconstruct the text using Vec2Word, as shown in Figure 10b. We observe that the approximated text is almost the same as the original one even with only $1/5$ principal vectors (difference highlighted in bold red).

11. Model and Training Details

In this appendix, we provide the model architectures and the hyperparameters of the experiments presented in Section 6.



(a) Singular values in word embedding vectors from BERT and Word2Vec.

Large Language Models are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. Large Language Models can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that **is** coherent and grammatically accurate.

(b) Original text

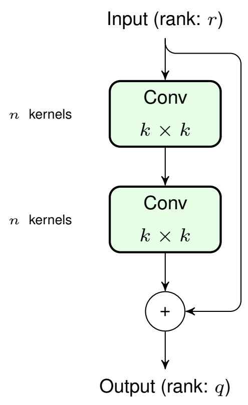
Large Language Models are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. Large Language Models can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that **are** coherent and grammatically accurate.

(c) approximated text with $1/5$ principal vectors from Word2Vec.

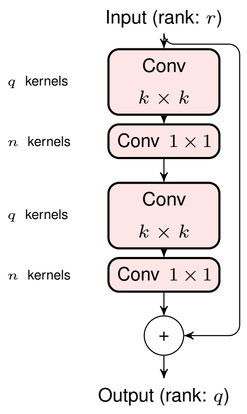
Figure 10. Asymmetric structure in language models. Embeddings in language models also have a highly asymmetric structure. An approximated text with only $1/5$ principal vectors is almost the same as the original.

11.1. Model Architectures

Since the backbone model $\mathcal{M}_{\text{bb}}$ and high-dimensional model $\mathcal{M}_{\text{res}}$ combined is just the original model, in this section, we omit their architecture details and only provide the architecture of $\mathcal{M}_{\text{main}}$.



(a) Original *Resblock*(n, k)



(b) Low-dim *Resblock*(n, k, q)

Figure 11. The original *Resblock* and low-dimensional *Resblock*. Non-linear activation functions and batchnorm are not shown for simplicity.

The original and low-dimensional *Resblocks* are shown in Figure 11. The design of the low-dimensional *Resblock* follows Figure 4 and Theorem 1.

Table 9 lists the details of model parameters of $\mathcal{M}_{\text{main}}$

	CIFAR10: $\epsilon = 1.4$	$\epsilon = \infty$	CIFAR100: $\epsilon = 1.4$	$\epsilon = \infty$
With Quantization	92.4 ± 0.3	93.7 ± 0.3	71.4 ± 0.1	74.8 ± 0.3
No Quantization	92.8 ± 0.2	94 ± 0.2	72 ± 0.4	75.1 ± 0.2

Table 7. Ablation study w/ and w/o quantization with ResNet-18. This demonstartes that the quantization does not significantly reduce model performance.

Table 8. Hyperparameters in investigating asymmetric structures in CNNs

batch size	epochs	lr	wd	momem.	lr scheduler
128	100	0.1	1e-4	0.9	cosine anneal

Table 9. Model parameters of $\mathcal{M}_{\text{main}}$ for ResNet-18 and ResNet-34 with 8 principal channels in IR_{main}.

ResNet-18				ResNet-34			
Resblock	n	k	q	*Resblock*	n	k	q
1,2	64	3	16	1-3	64	3	16
3,4	256	3	32	4-9	256	3	32
5,6	512	3	64	10-11	512	3	64

for ResNet-18 and ResNet-34. Given a *Resblock* with 3×3 kernels and input rank r, an output with rank $2r$ is sufficient to preserve most information in the principal channels [41]. Hence, for ResNet-18 and ResNet-34, we let $q = 2r$.

11.2. Hyperparameters in the Main Experiments

For the privacy parameters, we set δ as 10^{-6} for all datasets. Table 10 and 11 list hyperparameters in training ResNet-18 on CIFAR-10/100, and ResNet-18/34 on ImageNet[1].

Table 10. Hyperparameters in training ResNet-18 on CIFAR-10/100.

epochs	b	lr	wd	`orth reg`	r	t/t'
150	64	0.1	2e-4	8e-4	8	16/8

b: batch size, lr: initial learning rate, wd: weight decay.
r: #principal channels in IR_{main}, t/t': DCT/IDCT block sizes.
`orth reg`: kernel orthogonalization regularization.

Table 11. Hyperparameters in training ResNet-18/34 on ImageNet.

epochs	b	lr	wd	`orth reg`	r	t/t'
100	256	0.1	2e-5	0	12	14/7

12. More Related Works

In addition to the prior privacy-preserving machine learning works mentioned in the main paper, there are other related works in the current literature.

Split Learning. Split learning [53, 54, 57] is another training framework targeting data protection when sharing data with other parties. It splits and distributes a full model between private clients and untrusted public servers. During training, the clients learn a few front layers and send intermediate representations (rather than raw inputs) to the server, relieving computation and memory pressure on clients.

However, even the intermediate representations still contain substantial sensitive information about the raw input data, which gives way to adversaries who can infer training data, especially using a model inversion attack (Section 6.3). While split learning can be combined with DP to ensure privacy of the intermediate representations as in [54], unfortunately, this leads to a significant accuracy drop.

Crypto-based Private Learning. Privacy-preserving machine learning enhanced by crypto techniques provides strong data protection [22, 31, 32, 51]. These approaches first encrypt the input data and directly train a model in the encrypted domain, preventing any untrusted parties from obtaining raw data. However, the encryption/decryption and bootstrapping [2] operations add tremendous complexities during training and inference, limiting their use for large-scale models. Moreover, as non-linear activation functions are usually not supported by current encryption schemes, the crypto based solutions need to approximate these functions, which inevitably causes performance degradation.

TEE-based Private Learning. Trusted execution environments (TEEs) provide a secure hardware enclave for sensitive data, which makes it a practical option for privacy-preserving machine learning [14, 19, 26, 46, 47]. TEE-based solutions encapsulate the data and the models in a secure environment and perform forward and backward passes. Throughout the whole process, the private information is always secured in TEEs. Therefore, such solutions achieve strong privacy protection for both data and the model. One critical concern of using TEEs for machine learning, however, is the relatively low computing performance compared to GPUs. Due to the low parallelism and low communication efficiency, training/inference time with TEEs differs from that with GPUs by a factor of about 100 [41]. While the most advanced GPUs also come with trusted environment [43], it still remains to see how this can be applied in real large-scale applications.

[1]The DCT block size is chosen by trading off DCT computation complexity and the effectiveness of the low-frequency approximation. DCT with too small blocks does not effectively extract the low-frequency components, whereas larger block sizes are computationally-intensive.