Kernel Normalized Convolutional Networks for Privacy-Preserving Machine Learning

Reza Nasirigerdeh, Javad Torkzadehmahani, Daniel Rueckert, Georgios Kaissis

Abstract—Normalization is an important but understudied challenge in privacy-related application domains such as federated learning (FL), differential privacy (DP), and differentially private federated learning (DP-FL). While the unsuitability of batch normalization for these domains has already been shown, the impact of other normalization methods on the performance of federated or differentially private models is not well-known. To address this, we draw a performance comparison among layer normalization (LayerNorm), group normalization (GroupNorm), and the recently proposed kernel normalization (KernelNorm) in FL, DP, and DP-FL settings. Our results indicate LayerNorm and GroupNorm provide no performance gain compared to the baseline (i.e., no normalization) for shallow models in FL and DP. They, on the other hand, considerably enhance the performance of shallow models in DP-FL and deeper models in FL and DP. KernelNorm, moreover, significantly outperforms its competitors in terms of accuracy and convergence rate (or communication efficiency) for both shallow and deeper models in all considered learning environments. Given these key observations, we propose a kernel normalized ResNet architecture called KNResNet-13 for differentially private learning. Using the proposed architecture, we provide new state-of-the-art accuracy values on the CIFAR-10 and Imagenette datasets, when trained from scratch.

Index Terms—Differential Privacy, Federated Learning, Kernel Normalization, Group Normalization, Batch Normalization

I. INTRODUCTION

Deep convolutional neural networks (CNNs) are popular in a diverse range of image vision tasks including image classification [1]. Deep CNNs rely on large-scale datasets to effectively train the model, which might be difficult to provide in a centralized manner [2]. This is because datasets are often distributed across different sites such as hospitals, and contain sensitive data which cannot be transferred to a centralized location due to privacy regulations [3]. Even if such datasets become available, training algorithms can pose privacy risks to the individuals participating in the dataset, leaking privacy-sensitive information through the trained model [4]–[6].

Federated learning (FL) [7] addresses the large-scale data availability challenge by enabling clients to jointly train a global model under the coordination of a central server without sharing their private data. Network communication, on the other hand, emerges as a new challenge in federated environments, requiring a large number of communication rounds for model convergence, and exchanging a large amount of traffic in each round [8]. FL also causes utility (e.g. in terms of accuracy) reduction due to the non-IID (not independent and identically distributed) nature of the data across the clients [9]. Finally, although FL eliminates the requirement of data sharing, it might still lead to privacy leakage, where the private data of the clients can be reconstructed from the model updates shared with the server [10]–[12].

Differential privacy (DP) [13] copes with the privacy challenge in both centralized and federated environments by injecting random noise into the model gradients to limit the amount of information learnt about a particular sample in the dataset [14]. DP, however, adversely affects the model utility similar to FL because of the injected noise. In general, there is a trade-off between privacy and utility in DP, where stronger privacy leads to lower utility [15].

Batch normalization (BatchNorm) [16] is the de facto normalization layer in popular deep CNNs such as ResNets [17] and DenseNets [18], which remarkably improves the model convergence rate and accuracy in centralized training. BatchNorm, however, is not suitable for FL and DP settings. This is because BatchNorm relies on the IID distribution of feature values in the batch [16], which is not the case in federated settings. Moreover, per-sample gradients are required to be computed in DP that is impossible for batch-normalized CNNs [14]. Batch-independent layers such as layer normalization (LayerNorm) [19], group normalization (GroupNorm) [20], and the recently proposed kernel normalization (KernelNorm) [21] do not suffer from the limitations of BatchNorm, and therefore, are applicable to federated and differentially private learning.
**Normalization challenge.** Unsuitability of BatchNorm for federated and differentially private learning has presented a real challenge in the corresponding environments. Unlike the other challenges (i.e., utility, network communication, and privacy), the normalization issue has remained understudied in the context of FL and DP. Previous works [9], [22] illustrate that GroupNorm outperforms BatchNorm in terms of accuracy in federated settings. Likewise, GroupNorm also delivers higher accuracy than LayerNorm in differentially private learning [23]–[25]. Additionally, KernelNorm achieves significantly higher accuracy and faster convergence rate compared to LayerNorm and GroupNorm in both FL and DP settings according to the original study [21].

However, the prior studies have not made a comparison between different normalization layers and the NoNorm (no normalization layer) case in the first place. Moreover, the experimental evaluation regarding FL and DP environments is limited in the original KernelNorm study [21], focusing on a cross-silo federated setting (few clients with relatively large datasets) [26] and a shallow model in DP. Finally, the performance comparisons in the previous works do not consider differentially private federated learning (DP-FL) settings. Given that, two fundamental questions arise: (1) Do LayerNorm, GroupNorm, and KernelNorm also deliver higher performance than NoNorm in FL, DP, and DP-FL environments?, and (2) Does KernelNorm still outperform other normalization layers in cross-device FL (many clients with small datasets), in DP-FL, and using deeper models in DP?

**Key findings.** We conduct extensive experiments using the VGG-6 [27], ResNet-8 [21], PreactResNet-18 [28], and DenseNet20×16 [18] models trained on the CIFAR-10/100 [29] and Imagenet [30] datasets in FL, DP, and DP-FL settings to address those questions. The findings are as follows:

1) LayerNorm and GroupNorm do not necessarily outperform the NoNorm case for shallow models in FL and DP settings. For instance, LayerNorm and GroupNorm provide slightly lower accuracy and communication efficiency than NoNorm in the cross-silo federated setting, where the shallow VGG-6 model is trained on CIFAR-10. Similarly, LayerNorm and GroupNorm achieve lower accuracy than NoNorm using the shallow ResNet-8 model on CIFAR-10 in DP (Section III).

2) KernelNorm significantly outperforms NoNorm, LayerNorm, and GroupNorm in terms of communication efficiency (convergence rate) and accuracy in both cross-silo and cross-device FL, with both shallow and deeper models in DP, and using shallow models in DP-FL environments (Section III).

**Solution.** Based on our findings, we advocate employing KernelNorm as the effective normalization layer for FL, DP, and DP-FL settings. Given that, we propose a KernelNorm-based ResNet architecture called KRNResNet-13, and show it delivers considerably higher accuracy than the state-of-the-art GroupNorm-based architectures on CIFAR-10 and Imagenet in differentially private learning environments (Section IV).

**Contributions.** We make the following contributions: (I) we show LayerNorm and GroupNorm do not deliver higher accuracy than NoNorm with shallow models in FL and DP settings, (II) we illustrate the recently proposed KernelNorm layer has a great potential to become the de facto normalization layer in privacy-enhancing/preserving machine learning, and (III) we propose the KRNResNet-13 architecture, and provide new state-of-the-art (SOTA) accuracy values on CIFAR-10 and Imagenet using the proposed architecture in DP environments, when trained from scratch.

**II. PRELIMINARIES**

**Federated learning (FL).** A federated environment consists of multiple clients as data holders and a central server as coordinator. FL is a privacy-enhancing technique, which enables the clients to train a global model without sharing their private data with a third party. In FL, or more precisely in the FEderatedAVeraging (FedAvg) algorithm [7], the server randomly chooses K clients, and sends them the global model parameters $W_i^g$ in each communication round $i$. Next, each selected client $j$ trains the global model on its local dataset using mini-batch gradient descent, and shares the local model parameters $W_i^l_j$ with the server. Finally, the server takes the weighted average over the local parameters from the clients to update the global model:

$$W_{i+1}^g = \frac{\sum_{j=1}^K N_j \cdot W_i^l_j}{\sum_{j=1}^K N_j},$$

where $N_j$ is the number of samples in client $j$.

A cross-device federated setting contains a large number of clients such as mobile devices with small datasets [26]. The server selects a fraction of clients in each round. Moreover, the underlying assumption is that the communication between clients and server is unstable, and the clients might drop out during training. A cross-silo setting, on the other hand, consists of few clients such as hospitals or research institutions with relatively large datasets and stable network connection [26]. All clients participate in model training in all communication rounds. For more details on federated learning, the readers are referred to [7] and [26].

**Differential privacy (DP).** The differential privacy approach provides a theoretical framework and collection of techniques for privacy-preserving data processing and release [13]. Its guarantees are formulated in an information-theoretic fashion and describe the upper bound on the multiplicative information gain of an adversary observing the output of a computation over a sensitive database. This definition endows DP with a robust theoretical underpinning and ascertains that its guarantees hold in the presence of adversaries with unbounded prior knowledge and under infinite post-processing. Moreover, DP guarantees are compositional, meaning that they degrade predictably when a DP system is executed repeatedly on the same database. Formally, a randomised mechanism $\cal M$ is said to preserve $(\varepsilon, \delta)$-DP if, for all databases $D$ and $D'$
differing in the data of one individual and all measurable subsets \( S \) of the range of \( \mathcal{M} \), the following inequality holds:

\[
P(\mathcal{M}(D) \in S) \leq e^{\epsilon}P(\mathcal{M}(D') \in S) + \delta,
\]

where \( P \) is the probability of an event, \( \epsilon \geq 0 \) and \( 0 \leq \delta < 1 \). Of note, this inequality must hold also if \( D \) and \( D' \) are swapped. The guarantee is given over the randomness of \( \mathcal{M} \).

Intuitively, this characterisation implies that the output of the mechanism should not change too much when one individual’s data is added or removed from a database, or equivalently, the influence of one individual’s data on the result of the computation should be small.

The application of DP to the training of neural networks is usually (and in our work) based on the differentially private stochastic gradient descent (DP-SGD) algorithm [14]. Here, the role of the database is played by the individual (per-sample) gradients of the loss function with respect to the parameters. For the DP guarantee to be well-defined, the intermediate layer outputs (activations), leading to the computation of a per-sample gradient, are not allowed to be influenced by more than one sample. Hence, layers like BatchNorm, which normalize the activations of a layer by considering either other samples in the batch or the statistics of previously seen batches, cannot be employed in DP. We refer the readers to [13], [14], [31] for more information on differential privacy.

Differentially private federated learning (DP-FL). Although FL enhances data privacy by eliminating the requirement of data sharing, the model parameters shared with the server can still cause privacy leakage. To overcome this problem, the clients can rely on DP to train the global model on their local data, and share differentially private models with the server. This way, the clients can benefit from the guarantees of DP in federated environments.

Normalization. The normalization layers play a crucial role in deep CNNs. They can smoothen the optimization landscape [32] and effectively address the problem of vanishing gradients [33], leading to improved model performance. The normalization layers are different from each other in their normalization unit, which is a subset of elements from the original input that are normalized together with the mean and variance of the unit [21]. Assume that the input is a 4-dimensional tensor with parameters. For the DP guarantee to be well-defined, the intermediate layer outputs (activations) are normalized separately for each sample in the batch. The normalization unit of GroupNorm, however, is a tensor of shape \((c, k_h, k_w)\), where \(c\) is the number of input channels, and \((k_h, k_w)\) is the kernel size. Thus, KernelNorm considers all elements in the channel dimension but a subset of elements specified by the kernel size from the height and width dimensions during normalization. In simple words, KernelNorm is similar to the pooling layers, except that KernelNorm normalizes the elements instead of computing average or maximum, and carries out operation over all channels rather than on a single channel.

Formally, KernelNorm (1) applies dropout to the original normalization unit \( U \) to obtain the dropped-out unit \( U' \), (2) calculates the mean and variance of \( U' \), and (3) employs the computed mean and variance to normalize \( U \):

\[
U' = D_p(U),
\]

\[
\mu_{u'} = \frac{1}{c \cdot k_h \cdot k_w} \sum_{i_c=1}^{c} \sum_{i_h=1}^{k_h} \sum_{i_w=1}^{k_w} U'(i_c, i_h, i_w),
\]

\[
\sigma_{u'}^2 = \frac{1}{c \cdot k_h \cdot k_w} \sum_{i_c=1}^{c} \sum_{i_h=1}^{k_h} \sum_{i_w=1}^{k_w} (U'(i_c, i_h, i_w) - \mu_{u'})^2,
\]

\[
\hat{U} = \frac{U - \mu_{u'}}{\sqrt{\sigma_{u'}^2 + \epsilon}},
\]

where \( p \) is the dropout [35] probability, \( \mu_{u'} \) and \( \sigma_{u'}^2 \) are the mean and variance of \( U' \), respectively, and \( \hat{U} \) is the normalized unit. Partially inspired by BatchNorm, KernelNorm introduces a regularizing effect during training through normalizing the elements of the original unit \( U \) via the statistics calculated over the dropped-out unit \( U' \).

KernelNorm is a local normalization layer. Moreover, it has no learnable parameters, and its output might have very different shape than the input. Similar to LayerNorm and GroupNorm, KernelNorm is batch-independent because it performs normalization separately for each sample of the batch. The kernel normalized convolutional (KNCConv) layer [21] is the combination of the KernelNorm and convolutional layer, where the output of the former is given as input to the latter.

The modern CNNs are batch-normalized, leveraging the BatchNorm and convolutional layers in their architectures. The corresponding layer/group-normalized networks are obtained by simply replacing BatchNorm with LayerNorm/GroupNorm. The kernel-normalized counterparts [21], on the other hand, employ the KernelNorm and KNCConv layers as the main building blocks, while forgoing the BatchNorm layers. For more details on the normalization layers, the readers can see [16], [19]–[21].
We conduct extensive experiments to investigate the performance of different batch-independent normalization layers including LayerNorm, GroupNorm, and KernelNorm in the cross-silo and cross-device FL as well as DP and DP-FL environments. In the following, we first provide the description of the datasets, models, and case studies, and then discuss the results and findings.

A. Experimental Setup

Datasets. The CIFAR-10/100 dataset [29] contains 50000 train and 10000 test samples of shape $32 \times 32$ from 10/100 classes. The Imagenette dataset (160-pixel version) [30] is a subset of ImageNet [36], including 9469 train and 3925 validation images from 10 "easily classified" labels. The feature values are divided by 255 for KernelNorm based models, whereas they are normalized using the mean and standard deviation of CIFAR-10/100 or ImageNet for NoNorm, LayerNorm, and GroupNorm based counterparts. The samples of Imagenette are resized to $128 \times 128$.

Models. We adopt the VGG-6 architecture from [27], ResNet-8 model from [21], PreactResNet-18 implementation from [37], and DenseNet-20×16 (depth of 20 and growth rate of 16) implementation from [38]. In layer/group-normalized networks, BatchNorm is substituted by LayerNorm/GroupNorm. In the NoNorm case, the BatchNorm layers are either removed or replaced with the identity layer. The kernel-normalized counterparts are implemented by removing the BatchNorm layers, replacing the convolutional layers with KNConv, and inserting a KernelNorm layer before the final average-pooling layer in the ResNet, PreactResNet, and DenseNet models. In FL, the models employ the ReLU activation. In DP, on the other hand, the activation function is Mish [39], which was successfully used in [24] to achieve SOTA accuracy. We implement the models in the PyTorch library (version 1.1.1) [40].

Case Studies. We design nine different case studies (four in FL, three in DP, and two in DP-FL) to make the performance comparison among the normalization layers:

1) CIFAR-10-VGG-6 (cross-silo FL): This case study aims to train the shallow VGG-6 model on the low-resolution CIFAR-10 dataset in a cross-silo federated environment containing 10 clients, where each client has samples from only 2 classes. The sample sizes of the clients are almost the same.

2) CIFAR-10-VGG-6 (cross-device FL): Similar to the cross-silo counterpart, but in a cross-device federated setting including 100 clients, where 20 clients are randomly selected in each round.

3) CIFAR-100-PreactResNet-18 (cross-silo FL): The aim of this case study is to train the deeper PreactResNet-18 model on more challenging, low-resolution CIFAR-100 dataset in a cross-silo federated environment consisting of 10 clients with samples from 20 labels. The clients have highly similar sample sizes.

4) CIFAR-100-PreactResNet-18 (cross-device FL): Akin to the cross-silo counterpart, but in a cross-device federated setting consisting of 100 clients, where 20 clients are randomly chosen by the server in each round.

5) CIFAR-10-ResNet-8 (DP): The goal of this case study is to train the shallow ResNet-8 model on the low-resolution CIFAR-10 dataset in the DP environment.

6) CIFAR-10-DenseNet-20×16 (DP): This case study aims to train the deeper DenseNet-20×16 model on the low-resolution CIFAR-10 dataset in the DP setting.

7) Imagenette-PreactResNet-18 (DP). The purpose of this case study is to train the deeper PreactResNet-18 model on the medium-resolution Imagenette dataset in the differentially private environment.

8) CIFAR-10-VGG-6 (DP-FL): This case study aims to train the VGG-6 model on the CIFAR-10 dataset in a differentially private federated setting with 10 clients, where the clients have samples from 4 classes. The sample sizes of the clients are highly similar.

9) CIFAR-10-ResNet-8 (DP-FL): Similar to the previous case study, but with ResNet-8 as the model.

Federated training. We employ five different values for learning rate tuning in the federated case studies: $\eta = \{0.005, 0.01, 0.025, 0.05, 0.1\}$. The KernelNorm based models are trained for 400 and 1000 communication rounds in the CIFAR-10 and CIFAR-100 case studies, respectively. The number of rounds for the NoNorm, LayerNorm, and GroupNorm based models is as twice as the kernel normalized counterparts due to their slower convergence rate. The group size is the default value of 32 for the GroupNorm layer [20]. The dropout probability for KNConv and KernelNorm layers are 0.1 and 0.5, respectively. The loss function is cross-entropy, optimizer is SGD with momentum of zero, and training algorithm is FedAvg with number of local epochs of 1.

Differentially private training. We set $\epsilon = 6.0$ and $\delta = 10^{-5}$ for all DP case studies. Regarding parameter tuning, we use learning rate values of $\eta = \{1.0, 1.5, 2.0\}$ and clipping values of $C = \{1.0, 1.5, 2.0\}$. The ResNet-8, DenseNet-20×16, and PreactResNet-18 models are trained for 50, 70, and 70 epochs, respectively. The learning rate is divided by 2 at epochs (T-30) and (T-10), where T is the number of epochs (i.e. 50 or 70). The group size of GroupNorm is 16 for DenseNet-20×16, but 32 for the other models. Notice that we cannot set group size to 32 for DenseNet-20×16 because the number of channels must be divisible by the group size. The dropout probability is 0.1 for all KNConv layers in the kernel normalized models. For ResNet-8, the dropout probability of KernelNorm is 0.25, whereas it is 0.5 for DenseNet-20×16 and PreactResNet-18.

We employ cross-entropy as loss function, zero-momentum SGD as optimizer, and the Opacus library (version 1.1) [41] for model training. We observe that changing the kernel size of the shortcut connections in PreactResNet-18 from 1×1 to 2×2 slightly enhances the accuracy of the kernel normalized model, but provides no accuracy gain for the competitors. Thus, the aforementioned kernel size remains 1×1 for NoNorm, LayerNorm, and GroupNorm, whereas it is 2×2 for KernelNorm.
TABLE I: **Federated learning**: Test accuracy for different normalization layers; NoNorm (no normalization) slightly outperforms LayerNorm and GroupNorm in (a); KernelNorm delivers higher accuracy than the competitors; B: batch size.

(a) CIFAR-10-VGG-6 *(cross-silo FL)*

| B | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---|--------|-----------|-----------|------------|
| 16 | 80.19±0.29 | 78.93±0.43 | 78.63±0.56 | **83.64±0.41** |
| 64 | 79.23±0.31 | 78.97±0.36 | 79.49±0.38 | **82.13±0.25** |

(b) CIFAR-10-VGG-6 *(cross-device FL)*

| B | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---|--------|-----------|-----------|------------|
| 16 | 80.95±0.27 | 81.89±0.32 | 81.39±0.47 | **84.13±0.26** |
| 64 | 80.72±0.06 | 81.43±0.19 | 81.44±0.18 | **83.77±0.11** |

(c) CIFAR-100-PreactResNet-18 *(cross-silo FL)*

| B | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---|--------|-----------|-----------|------------|
| 16 | 61.89±0.13 | 68.16±0.44 | 67.86±0.10 | **71.72±0.19** |
| 64 | 60.89±0.33 | 66.90±0.41 | 66.45±0.18 | **71.29±0.21** |

(d) CIFAR-100-PreactResNet-18 *(cross-device FL)*

| B | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---|--------|-----------|-----------|------------|
| 16 | 63.54±0.22 | 68.05±0.92 | 68.23±0.13 | **71.75±0.24** |
| 64 | 63.33±0.36 | 67.84±0.43 | 67.47±0.24 | **71.99±0.09** |

Fig. 1: **Federated learning**: Communication efficiency for various normalization layers; KernelNorm provides significantly higher communication efficiency than the competitors. Surprisingly, NoNorm outperforms both LayerNorm and GroupNorm in terms of communication efficiency in most cases, i.e (a), (b), (d); batch size is 64.

**Differentially private federated training.** We set $\varepsilon=8.0$ and $\delta=10^{-5}$ for both DP-FL case studies. We leverage learning rate values of $\eta=\{0.01, 0.025, 0.05\}$ and clipping values of $C=\{1.0, 1.5, 2.0\}$ for parameter tuning. The group size of GroupNorm is 32, and the dropout probabilities of the KNConv and KernelNorm layers are 0.1 and 0.25, respectively. The models are trained for 100 communication rounds with a fixed learning rate. The loss function, optimizer, and training algorithm are cross-entropy, SGD with momentum of zero, and FedAvg with number of local epochs of 1, respectively.

**B. Results**

For all case studies, we first determine the optimal learning rate (and clipping value) based on the model accuracy on the test dataset (see Appendix). We repeat the experiment achieving the highest accuracy three times and report mean/median/mean and the standard deviation of the runs for the FL/DP/DP-FL case studies. We consider the average over the last 10 communication rounds, final accuracy, and the average over the last 3 rounds as the representative accuracy of the run in the FL, DP, and DP-FL settings, respectively.
TABLE II: Differential privacy: Test accuracy for various normalization layers; NoNorm (no normalization) delivers slightly higher accuracy than LayerNorm and GroupNorm in (a); KernelNorm considerably outperforms the competitors; $\varepsilon=6.0$, $\delta=10^{-5}$.

(a) CIFAR-10-ResNet-8 (DP)

| B      | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| 512    | 65.11±0.29 | 70.01±0.19 | 70.27±0.08 | **72.18±0.15** |
| 1024   | 69.05±0.4  | 71.38±0.5  | 71.75±0.45 | **74.31±0.14** |
| 2048   | 72.7±0.25  | 71.67±0.42 | 71.73±0.31 | **75.46±0.34** |
| 3072   | 71.99±0.14 | 69.39±0.27 | 68.99±0.27 | **75.48±0.24** |

(b) CIFAR-10-DenseNet-20×16 (DP)

| B      | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| 512    | 57.03±0.48 | 65.62±0.7  | 66.16±0.56 | **68.49±0.24** |
| 1024   | 64.15±0.74 | 69.24±0.68 | 68.72±0.65 | **70.86±0.44** |
| 2048   | 64.98±0.6  | 69.68±0.8  | 69.57±0.97 | **72.74±0.34** |
| 3072   | 65.29±0.53 | 66.66±0.78 | 67.31±0.26 | **72.49±0.39** |

(c) Imagenette-PreactResNet-18 (DP)

| B      | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| 512    | 25.27±3.95 | 54.83±0.65 | 56.7±0.19  | **59.1±0.33** |
| 1024   | 53.69±0.83 | 54.54±0.23 | 57.17±0.42 | **58.9±0.42** |
| 2048   | 53.53±0.99 | 53.3±0.32  | 54.59±0.27 | **56.11±0.26** |

Fig. 2: Differential privacy: Convergence rate for different normalization layers; kernel normalized models provides much faster convergence rate than the competitors; batch size is 2048, 1024, and 1024 for (a), (b), and (c), respectively.
TABLE III: Differentially private federated learning: Test accuracy for different normalization layers; KernelNorm delivers considerably higher accuracy than the competitors; $\varepsilon=8.0$, $\delta=10^{-5}$.

|               | B   | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---------------|-----|--------|-----------|-----------|------------|
|               | 256 | 30.5±0.44 | 38.23±0.37 | 37.29±0.71 | 46.79±0.81 |
|               | 512 | 29.73±1.01 | 39.47±0.48 | 39.75±0.65 | 45.37±0.22 |
|               | 1024| 33.43±1.33 | 39.19±0.64 | 38.85±0.97 | 47.11±0.37 |

(a) CIFAR-10-VGG-6 (DP-FL)  (b) CIFAR-10-ResNet-8 (DP-FL)

|               | B   | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|---------------|-----|--------|-----------|-----------|------------|
|               | 256 | 34.76±0.95 | 38.43±1.48 | 40.69±1.03 | 45.18±0.34 |
|               | 512 | 36.11±0.7  | 41.09±0.33 | 41.8±0.41  | 46.75±0.48 |
|               | 1024| 38.19±0.19 | 41.41±1.08 | 41.39±0.82 | 48.45±1.09 |

Federated learning. Table I lists the test accuracy values for the FL case studies. According to the table, (1) NoNorm slightly outperforms LayerNorm and GroupNorm in the CIFAR-10-VGG-6 (cross-silo FL) case study, whereas LayerNorm and GroupNorm deliver higher accuracy compared to NoNorm in the other case studies; (2) KernelNorm achieves considerably higher accuracy than the competitors. Fig. 1 illustrates the communication efficiency (i.e. accuracy versus communication round) for the FL case studies. As shown in the figure, (1) NoNorm, surprisingly, provides higher communication efficiency than LayerNorm and GroupNorm for most case studies; (2) KernelNorm achieves remarkably higher communication efficiency compared with NoNorm, LayerNorm, and GroupNorm.

Differential privacy. Table II and Fig. 2 demonstrate the test accuracy and convergence rate of different normalization layers for the DP case studies, respectively. According to the table and figure, (1) NoNorm slightly outperforms LayerNorm and GroupNorm in terms of accuracy in the CIFAR-10-ResNet-8 (DP) case study, but LayerNorm and GroupNorm achieve higher accuracy compared to NoNorm in the other case studies, (2) KernelNorm provides higher accuracy than the competitors in all DP case studies, and (3) KernelNorm based models converge much faster than those based on NoNorm, LayerNorm, and GroupNorm.

Differentially private federated learning. Table III lists the test accuracy values, and Fig. 3 illustrates the convergence rate of different normalization layers for the DP-FL case studies. As shown in the table and figure, (1) the NoNorm based models deliver much lower accuracy and slower convergence rate than LayerNorm, GroupNorm, and KernelNorm based ones, and (2) the kernel normalized models achieve considerably higher accuracy and faster convergence rate than the competitors.

C. Findings

Based on our experimental evaluation, (I) LayerNorm and GroupNorm do not necessarily outperform NoNorm in shallow networks such as VGG-6/ResNet-8 under the FL/DP settings. However, they achieve significant accuracy gain compared to NoNorm for deeper models (e.g. DenseNet-20×16 and PreactResNet-18) in FL and DP as well as shallow models in DP-FL, and (II) KernelNorm delivers remarkably higher accuracy and convergence rate (communication efficiency) than NoNorm, LayerNorm, and GroupNorm with both shallow and deeper networks trained in FL (cross-silo and cross-device) and DP as well as shallow models in DP-FL. Therefore, KernelNorm is the most effective normalization method for FL, DP, and DP-FL settings.

IV. KERNEL NORMALIZED RESNET-13

The experimental results from the previous section indicate KernelNorm outperforms the competitors in the DP setting using models that originally designed based on global normalization layers such as BatchNorm (e.g. PreactResNets or DenseNets). The existing architectures, however, are not necessarily optimal for KernelNorm. For instance, the kernel size of 1×1 in the shortcut connections of the ResNet architecture is not beneficial for KernelNorm, which requires kernel sizes greater than 1 to benefit from the spatial correlation of the elements during normalization.
Given that, we propose a bespoke ResNet architecture for KernelNorm (Fig. 4) to improve the SOTA accuracy values on the CIFAR-10 and Imagenette datasets in differentially private learning settings. We refer to the proposed architecture as KNResNet-13, which includes twelve kernel normalized convolutional layers and a final classification (linear) layer.

The convolutional blocks in KNResNet-13 are either residual (Fig. 4a) or transitional (Fig. 4b). The residual blocks contain two KNConv layers with the same number of input and output channels. The transitional blocks include a KNConv and max-pooling layer, aiming to downsample the input. All KNConv layers have kernel size $3 \times 3$, stride $1 \times 1$, padding $1 \times 1$, and dropout probability 0.1. The kernel size of the max-pooling layers is $2 \times 2$. The architecture employs Mish as the activation function. The last residual block is followed by a KernelNorm layer with dropout probability 0.5, Mish activation, $2 \times 2$ adaptive average-pooling, and linear layer with 1024 neurons. For medium-resolution images (e.g., $224 \times 224$), the first KNConv layer is replaced by a $7 \times 7$ KNConv layer followed by the Mish activation and $2 \times 2$ max-pooling layer.

In the following, we describe the data preprocessing and differentially private training procedure for the CIFAR-10 and Imagenette datasets. Then, we provide the accuracy values achieved by the KNResNet-13 model and compare them with those from the recent studies.

**CIFAR-10.** The only data preprocessing step is to divide the feature values by 255. KNResNet-13 is trained for $T = 50, 70, 70,$ and 80 epochs with batch sizes of $B = 4096, 4096, 3072,$ and 3072 for $\varepsilon = 2.0, 4.0, 6.0,$ and 8.0, respectively. The learning rate is 2.0, clipping value is 1.5, and $\delta$ is $10^{-5}$. The learning rate is divided by 2 at epochs (T - 30) and (T - 10). The optimizer is SGD with momentum of zero.

**CIFAR-10 with augmentation multiplicity.** The augmentation multiplicity is a recently proposed technique by De et al. [23], which computes the gradients for a given sample by taking average over the gradients computed for different augmentations of the same sample. For the CIFAR-10 dataset, this technique applies the sequence of random horizontal flipping and random cropping of size $32 \times 32$ and padding $4 \times 4$ to obtain an augmented version of a given sample. Here, we employ a slightly different way of augmentation multiplicity because the original technique provides negligible accuracy gain for our model. We first compute the gradients for the original sample, horizontally flipped (i.e., with probability of 1.0), and randomly cropped version of the sample, and then take the average over them to calculate the per-sample gradients. For $\varepsilon = 2.0, 4.0, 6.0,$ and 8.0, KNResNet-13 is trained for 80, 80, 100, and 100 epochs, respectively. The other training details are the same as CIFAR-10 with no augmentation multiplicity (previous paragraph).

**Imagenette.** We adopt the 320-pixel version of the dataset and resize the images to $224 \times 224$. We train KNResNet-13 with $\eta = 1.5, C = 1.5, \varepsilon = 7.0, \delta = 10^{-5}$, and zero-momentum SGD for 100 epochs, where $\eta$ is divided by 2 at epochs 70 and 90.

**Results.** Table IV lists the test accuracy values from KNResNet-13 and the recent studies on CIFAR-10, CIFAR-10 with augmentation multiplicity, and Imagenette. KNResNet-13 delivers significantly higher accuracy than the models based on GroupNorm or NoNorm for all considered $\varepsilon$ values on CIFAR-10 without augmentation multiplicity. Compared to kernel normalized ResNet-8 [21], KNResNet-13 provides up to 2% accuracy gain depending on the $\varepsilon$ value.

On CIFAR-10 with augmentation multiplicity, KNResNet-13 outperforms both wide ResNet-16-4 and ResNet-40-4 [43]
### Table IV: Differential privacy

Comparison of the test accuracy values from the proposed KNResNet-13 architecture with those from the recent studies; $\delta=10^{-5}$.

**(a) CIFAR-10**

| Study               | Model      | Normalization | $\epsilon$ | Test accuracy |
|---------------------|------------|---------------|-------------|---------------|
| Klause et al. (2022) [24] | ResNet-9   | GroupNorm     | 9.88        | 73.0          |
| Nasirigerdeh et al. (2022) [21] | ResNet-8   | KernelNorm    | 8.0         | 76.66         |
| Ours                | KNResNet-13| KernelNorm    | 8.0         | **78.51±0.35**|
| Dörmann et al. (2021) [42] | VGG-8      | NoNorm        | 7.42        | 70.1          |
| Klause et al. (2022) [24] | ResNet-9   | GroupNorm     | 7.42        | 71.8          |
| Remerscheid et al. (2022) [25] | DenseNet-14| GroupNorm     | 7.0         | 73.5          |
| Nasirigerdeh et al. (2022) | ResNet-8   | KernelNorm    | 6.0         | 75.46         |
| Ours                | KNResNet-13| KernelNorm    | 6.0         | **77.09±0.31**|

**(b) CIFAR-10 with augmentation multiplicity (K)**

| Study               | Model      | Normalization | K  | $\epsilon$ | Test accuracy |
|---------------------|------------|---------------|----|-------------|---------------|
| De et al. (2022) [23] | Wide ResNet-16-4| GroupNorm     | 16 | 8.0         | 79.5          |
| De et al. (2022) [23] | Wide ResNet-40-4| GroupNorm     | 32 | 8.0         | **81.4**      |
| Ours                | KNResNet-13| KernelNorm    | 3  | 8.0         | 80.8±0.22     |
| De et al. (2022) [23] | Wide ResNet-16-4| GroupNorm     | 16 | 6.0         | 77.0          |
| De et al. (2022) [23] | Wide ResNet-40-4| GroupNorm     | 32 | 6.0         | 78.8          |
| Ours                | KNResNet-13| KernelNorm    | 3  | 6.0         | **79.09±0.07**|
| De et al. (2022) [23] | Wide ResNet-16-4| GroupNorm     | 16 | 4.0         | 71.9          |
| De et al. (2022) [23] | Wide ResNet-40-4| GroupNorm     | 32 | 4.0         | 73.5          |
| Ours                | KNResNet-13| KernelNorm    | 3  | 4.0         | **76.19±0.04**|
| De et al. (2022) [23] | Wide ResNet-16-4| GroupNorm     | 16 | 2.0         | 64.9          |
| De et al. (2022) [23] | Wide ResNet-40-4| GroupNorm     | 32 | 2.0         | 65.9          |
| Ours                | KNResNet-13| KernelNorm    | 3  | 2.0         | **70.57±0.24**|

**(c) Imagenette**

| Study               | Model      | Normalization | $\epsilon$ | Test accuracy |
|---------------------|------------|---------------|-------------|---------------|
| Klause et al. (2022) [24] | ResNet-9   | GroupNorm     | 7.42        | 64.8          |
| Klause et al. (2022) [24] | ResNet-9   | GroupNorm     | 9.88        | 67.1          |
| Remerscheid et al. (2022) [25] | DenseNet-14| GroupNorm     | 7.0         | 69.7          |
| Ours                | KNResNet-13| KernelNorm    | 7.0         | **72.24±0.48**|

with much lower augmentation multiplicity (3 vs. 16 vs. 32) for $\epsilon$ values of 2.0, 4.0, and 6.0. On Imagenette, KNResNet-13 achieves around 3% and 7% higher accuracy than GroupNorm based DenseNet-14 [25] and ResNet-9 [24], respectively.

Given the results from Table IV, we provide new SOTA accuracy values on the CIFAR-10 and Imagenette datasets, when trained from scratch:

- On CIFAR-10 without augmentation multiplicity, the accuracy values of 74.51%, 77.09%, and 78.51% for $\epsilon=4.0$, 6.0, and 8.0, respectively.
- On CIFAR-10 with augmentation multiplicity, the accuracy values of 70.57%, 76.19%, and 79.09% for $\epsilon=2.0$, 4.0, and 6.0, respectively.
- On Imagenette, the accuracy value of 72.24% for $\epsilon=7.0$. 

115
Our experimental evaluation shows KernelNorm delivers higher performance than LayerNorm and GroupNorm in FL, DP, and DP-FL. This can be because KernelNorm is a local normalization method, taking into account the spatial correlation of the elements in the height and width dimensions during normalization. This leads to faster convergence rate compared to global batch-independent layers including LayerNorm and GroupNorm, likely due to the smoother optimization landscape [24]. It implies KernelNorm requires less amount of total injected noise to achieve a target accuracy value for a given privacy budget in DP, and a fewer number of communication rounds, and thus, higher communication efficiency in FL.

Moreover, LayerNorm and GroupNorm have scale and shift as learnable parameters. In FL, these parameters are aggregated, while they are perturbed with noise in DP. The performance of the layer and group normalized models can negatively be impacted in both cases. KernelNorm, however, is free of these learnable parameters, which can be another factor in superior performance of KernelNorm compared to LayerNorm and GroupNorm.

Finally, the feature values are not required to be normalized with the per-channel mean and standard deviation of the dataset in KernelNorm based models due to self-normalizing nature of KNConv, which normalizes the input before computing convolution. This is beneficial, especially in federated environments, because it is not required for clients to share the mean and standard deviation of their local datasets with the server to compute the corresponding global values.

Given the aforementioned properties and its superior performance, KernelNorm has a great potential to become the standard normalization layer for federated learning, differential privacy, and differentially private federated learning.

VI. RELATED WORK

There are few studies that compare the performance of various normalization layers in federated settings. Hsieh et al. [9] experimentally show GroupNorm delivers higher accuracy than BatchNorm in supervised FL. Zhang et al. [22] demonstrate this also holds for semi-supervised FL. However, these studies have not compared GroupNorm with NoNorm as the baseline. Our experiments illustrate GroupNorm does not necessarily provide accuracy gain compared to NoNorm for shallow models in supervised federated settings.

Several studies investigate the performance of different batch-independent normalization layers for differentially private learning. Klause et al. [24] and Remerscheid et al. [25] show GroupNorm outperforms LayerNorm in terms of accuracy in DP settings. Nasirigerdeh et al. [21] illustrate KernelNorm delivers considerable accuracy gain compared to both LayerNorm and GroupNorm in DP. These prior works, however, do not consider NoNorm as the baseline for comparison. Our evaluation indicates NoNorm slightly outperforms both LayerNorm and GroupNorm for the shallow ResNet-8 model on CIFAR-10, whereas KernelNorm still provides significant accuracy improvement compared to NoNorm for the aforementioned setting. The experimental evaluation of Nasirigerdeh et al. [21], moreover, is limited to a single case study. We conduct more extensive experiments with deeper models on both low-resolution and medium-resolution datasets to draw the performance comparisons among NoNorm, LayerNorm, GroupNorm, and KernelNorm.

Some studies propose novel architectures or data augmentation techniques to enhance the accuracy of differentially private models. Klause et al. [24] present a 9-layer ResNet architecture in which an additional normalization is performed after the addition operation of the residual block, and show their architecture improves the accuracy compared to the original ResNet architecture. Remerscheid et al. [25] introduce a novel DenseNet-based architecture called SmoothNet, which employs $3 \times 3$ convolutional layers with a high number of filters in the DenseNet blocks, and demonstrate it outperforms the previous ones in terms of accuracy. Both architectures employ GroupNorm as their normalization layer. We propose the KNResNet-13 architecture based on KernelNorm, and show it delivers considerably higher accuracy than the aforementioned architectures on CIFAR-10 and Imagenette.

De et al. [23] present the augmentation multiplicity technique, which computes the per-sample gradients by taking average over the gradients from different augmentations of the sample. We adopt this technique to train the proposed KNResNet-13 architecture on CIFAR-10. The accuracy from KNResNet-13 is higher than the wide ResNet-16-4 and ResNet-40-4 used in [23] for $\varepsilon$ values of 2.0, 4.0, and 6.0.

VII. CONCLUSION AND FUTURE WORK

We address the normalization challenge in the context of federated and differentially private learning. Through extensive experiments, we demonstrate: (1) in FL and DP, using no normalization layer in the architecture of shallow networks such as VGG-6 and ResNet-8 delivers slightly higher accuracy than LayerNorm and GroupNorm, (2) on deeper models such as DenseNet-20×16 and PreActResNet-18 in FL and DP as well as the shallow models in DP-FL, however, LayerNorm and GroupNorm considerably outperform NoNorm, and (3) the recently proposed KernelNorm method achieves significantly higher accuracy and convergence rate compared to NoNorm, LayerNorm, and GroupNorm in FL, DP, and DP-FL.

Given the superior performance of KernelNorm, we propose a kernel normalized ResNet architecture called KNResNet-13 for differentially private learning. Using the proposed architecture, we provide new SOTA accuracy values on CIFAR-10 with and without augmentation multiplicity as well as Imagenette for different $\varepsilon$ values, when trained from scratch.

We employ a low augmentation multiplicity value (i.e., 3) in our study due to the remarkable computational overhead of the technique. KNResNet-13 might deliver even higher accuracy with larger augmentation multiplicity values (e.g., 16 or 32), which can be an investigated in future studies. Additionally, the performance evaluation of kernel normalized architectures on the large Imagenet-32×32 dataset [36] is an interesting direction for future works.
### TABLE V: Federated learning
Learning rate values giving the highest accuracy for each normalization layer; B: batch size.

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 16     | 0.025  | 0.025     | 0.01      | 0.025      |
| 64     | 0.025  | 0.025     | 0.05      | 0.025      |

(a) CIFAR-10-VGG-6 (cross-silo FL)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 16     | 0.025  | 0.05      | 0.05      | 0.05       |
| 64     | 0.025  | 0.05      | 0.05      | 0.05       |

(b) CIFAR-10-VGG-6 (cross-device FL)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 16     | 0.01   | 0.01      | 0.005     | 0.025      |
| 64     | 0.01   | 0.01      | 0.01      | 0.05       |

(c) CIFAR-100-PreactResNet-18 (cross-silo FL)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 16     | 0.01   | 0.01      | 0.005     | 0.025      |
| 64     | 0.01   | 0.01      | 0.01      | 0.05       |

(d) CIFAR-100-PreactResNet-18 (cross-device FL)

### TABLE VI: Differential privacy
Learning rate values giving the highest accuracy for each normalization layer; B: batch size.

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 2.0    | 2.0       | 2.0       | 2.0        |
| 2048   | 2.0    | 2.0       | 2.0       | 2.0        |
| 3072   | 2.0    | 2.0       | 2.0       | 2.0        |

(a) CIFAR-10-ResNet-8 (DP)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 1.0    | 1.5       | 2.0       | 1.5        |
| 512    | 1.0    | 2.0       | 2.0       | 1.5        |
| 1024   | 1.5    | 2.0       | 1.5       | 1.5        |
| 2048   | 2.0    | 2.0       | 2.0       | 1.5        |

(b) CIFAR-10-DenseNet-20×16 (DP)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 1.0    | 1.0       | 1.0       | 2.0        |
| 2048   | 1.0    | 1.0       | 1.0       | 2.0        |

(c) Imagenette-PreactResNet-18 (DP)

### TABLE VII: Differential privacy
Clipping values giving the highest accuracy for each normalization layer; B: batch size.

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 1.0    | 1.5       | 2.0       | 1.5        |
| 2048   | 2.0    | 2.0       | 2.0       | 2.0        |
| 3072   | 2.0    | 2.0       | 2.0       | 2.0        |

(a) CIFAR-10-ResNet-8 (DP)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 1.0    | 1.5       | 2.0       | 1.5        |
| 512    | 1.0    | 2.0       | 2.0       | 1.5        |
| 1024   | 1.5    | 2.0       | 1.5       | 1.5        |
| 2048   | 2.0    | 2.0       | 2.0       | 1.5        |

(b) CIFAR-10-DenseNet-20×16 (DP)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 1.0    | 1.5       | 2.0       | 1.5        |
| 2048   | 1.0    | 1.0       | 1.0       | 2.0        |

(c) Imagenette-PreactResNet-18 (DP)

### TABLE VIII: Differentially private federated learning
Learning rates giving the highest accuracy for each norm layer.

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 0.01   | 0.01      | 0.01      | 0.01       |
| 512    | 0.025  | 0.01      | 0.01      | 0.025      |
| 1024   | 0.025  | 0.01      | 0.025     | 0.025      |

(a) CIFAR-10-VGG-6 (DP-FL)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 0.01   | 0.01      | 0.01      | 0.01       |
| 512    | 0.025  | 0.01      | 0.01      | 0.025      |
| 1024   | 0.025  | 0.01      | 0.025     | 0.025      |

(b) CIFAR-10-ResNet-8 (DP-FL)

### TABLE IX: Differentially private federated learning
Clipping values giving the highest accuracy for each norm layer.

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 1.0    | 1.5       | 1.0       | 1.0        |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 1.0    | 1.5       | 2.0       | 2.0        |

(a) CIFAR-10-VGG-6 (DP-FL)

|        | NoNorm | LayerNorm | GroupNorm | KernelNorm |
|--------|--------|-----------|-----------|------------|
| B      |        |           |           |            |
| 256    | 1.0    | 1.5       | 1.0       | 1.0        |
| 512    | 1.0    | 1.0       | 1.0       | 1.0        |
| 1024   | 1.0    | 1.0       | 2.0       | 2.0        |

(b) CIFAR-10-ResNet-8 (DP-FL)