We will explain communication more obviously in our revision. Although we get this takeaway in ResNet, “both” KD loss may be more effective to other models. So it is better to leave we do not use KD, sometimes it diverges, and sometimes it gets a low accuracy. We will clarify this in our revision.

Please note that the sizes of features and logits are fixed by the CNN architecture design. Compared to the entire model weight or gradient, the hidden vector is definitely much smaller (e.g., the hidden vector size of ResNet-110 is around 64KB while the entire gradient/model size is 4.6MB for 32x32 images). The hidden vector for each data point can be transmitted independently, thus GKT has less bandwidth requirement than gradient or model exchange, although the communication cost in total depends on the number of data points. Moreover, our experimental result shown in Figure 5 even demonstrates that our method has smaller communication costs for the entire training than split learning (a method that also exchanges hidden vectors during training) because of less communication round for convergence. We will explain communication more obviously in our revision.

Our method definitely can support the cross-device setting. First, we believe the user selection strategy is still an open problem. It is better to tailor the strategy for different models and optimization methods. The random subsampling method mentioned by R3 and R2 may not fit for our large DNN setting. The random subsampling may cause many users’ data to be touched only once. This is not a big issue for shallow NN because shallow NN requires much fewer data to converge than large DNN. However, it is a problem to large DNN because it typically requires all samples to be trained many epochs. Therefore, under GKT framework, it is more reasonable to use a pre-defined client selection strategy. All clients are divided into many groups. We then train group by group to make sure each group is trained multiple rounds (epochs). The optimizer state of each group can be maintained by uploading it to the server of GKT. Once the group ID is changed, the server then synchronizes the optimizer state to the clients in the new group. This training process is essentially the same as our GKT algorithms: from the perspective of optimization, both “viewing 10 users’ dataset as an epoch (as done in our experiments)” and “viewing each user” dataset as an epoch and training user by user” can converge. We will demonstrate this in our revision. Besides sampling, client-edge-cloud hierarchical FL is also a potential solution. We can use the edge server in the hierarchical topology to improve load balance. Also, from the perspective of alternating optimization, GKT does not have scalability issues since GKT does not do synchronous aggregation on the server-side: the server can immediately start training once it receives updates from any client.

Our work does not focus on privacy-preserving techniques, but please note that the hidden feature exchange happens at the training phase. This makes the attack harder because what the attacker access is the evolving and untrained feature map rather than a fully trained feature map that represents the raw data. Analyzing the degree of privacy leakage under our framework will be our future works.

We already discussed different knowledge distillation methods including FedMD in our related works. As for the language model, we extend it as a future work because careful consideration is needed to tailor for the characteristics of LSTM and Transformer (the back-propagation through time in LSTM and attention mechanisms in Transformer).