We thank all reviewers for their time and their valuable feedback. We will add corrections/clarifications as suggested. We would like to emphasize our contribution, as summarized by R5's thorough review: "What's novel about this paper is not the concept of applying knowledge distillation to FL or distributed training. Rather, the contribution of this paper is formulating a robust, efficient training scheme with extensive results and analysis which is significant enough."

**[R1: Misunderstanding on algorithm]** Our proposed FedDF is not the mentioned engineering solution. Each model architecture groups acquires knowledge from logits averaged over *all* received models (line 14 in Alg. 3, Fig. 7) for next FL round; thus mutual beneficial information can be shared *across architectures*. The $S_t$ at Line 13 in Alg. 3 is correct.

**[R1: No privacy concern in using GAN]** GAN training is not involved in all stages of FL and cannot steal clients' data. Data generation is done by the (frozen) generator *before the FL training* by performing inference on random noise.

**[R1: Clarity of line 32]** "[A]pplying ensemble learning techniques directly..." refers to keeping weights of all received models on the server and performing naive ensembling (logits averaging) for inference (line 30-31). In contrast, we distill the knowledge of *all* received models to the server model and then drop all received models' weights.

**[R2: Quality loss and data heterogeneity]** Final performance for all methods with different non-i.i.d. degrees/models/datasets is shown in Tab. 2 & 3, and FedDF is consistently the best performing method. As mentioned in the caption of Tab. 1, the fine-tuned test accuracy of centralized training (on all local data) is 86%. The data heterogeneity issue in FL results in quality loss (e.g. Fig. 2), and thus 80% and 75% are reasonable targets.

Due to computational infeasibility, ResNet-8 with fine-tuned hyper-parameters is used to provide in-depth empirical understanding. Better performance can be achieved by larger model capacity, but is orthogonal to the provided insights.

**[R2 & R4 & R5: Comments on preprints FedMD and Cronus]** We comment on the two closest approaches (FedMD and Cronus), in order to address 1) Distinctions between FedDF and prior work (R4), 2) Privacy/Communication traffic concerns (R2), 3) Omitted experiments on FedMD and Cronus (R2, R4, R5).

- Distinctions between FedDF and prior work. As discussed in the related work, most SOTA FL methods directly manipulate received model parameters (e.g. FedAvg/FedAvgM/FedMA). To our best knowledge, FedMD and Cronus are the only two that utilize logits information (of neural nets) for FL. The distinctions from them are made below.
- Different objectives and evaluation metrics. Cronus is designed for robust FL under poisoning attack, whereas FedMD is for personalized FL. In contrast, FedDF is intended for on-server model aggregation (evaluation on the aggregated model), whereas neither FedMD nor Cronus aggregates the model on the server.
- Different Operations. 1) FedDF, like FedAvg, *only* exchanges models between the server and clients (line 114), without transmitting input data. In contrast, FedMD and Cornus rely on exchanging public data logits. As FedAvg, FedDF can include privacy/security extensions and has the same communication cost per round. 2) FedDF performs ensemble distillation with unlabeled data *on the server*. In contrast, FedMD/Cronus use averaged logits received from the server for *local client training*.
- Omitted experiments with FedMD/Cronus. 1) FedMD requires to locally pre-train on the *labeled* public data, thus the model classifier necessitates an output dimension of # of public classes *plus* # of private classes (c.f. the output dimension of # of private classes in other FL methods). We cannot compare FedMD with FedDF with the same architecture (classifier) to ensure fairness. 2) Cronus is shown to be consistently worse than FedAvg in normal FL (i.e. no attack case) in their Tab. IV & VI. 3) We thoroughly evaluated SOTA baselines with the same objective/metric.

**[R4: Local training technique]** Our experimental setup is widely adopted in many other published FL papers. We observe that techniques like learning rate decay and local momentum are orthogonal to the model aggregation. Fig. 12 showcases the ineffectiveness of learning rate decay during local training. Including local Nesterov momentum only marginally improves all methods without affecting the conclusion; we will include omitted results for local momentum.

**[R5: Clarification on the "contradiction claims"]** FedMD and Cronus have no evaluations on their choices of training data construction, thus it remains unclear (line 89-90) how local training gets affected. Some general robust approaches reviewed in Cronus (e.g. Krum, Bulyan) can be adapted to exclude faulty client models for FedDF. These techniques alone do not interfere with the local training. We include extra results to justify the compatibility of FedDF with orthogonal work; *fine-tuned* proximal penalty (from FedProx) is used *locally* as suggested, on CIFAR-10 with ResNet-8 (setups in Fig. 2). For non-iid degree $\alpha = 1$, the results of FedDF v.s. FedAvg over three seeds are: w/ prox 80.56 v.s. 76.11 and w/o prox 80.27 v.s. 72.73; for $\alpha = 0.1$, we have w/ prox 71.64 v.s. 62.53, and w/o prox 71.52 v.s. 62.44.

**[R5: Learning rate]** Our learning rate tuning is actually sufficient, as our used STOA networks are much less sensitive to learning rate, different from the classical CNN (w/o BN and w/o residual connection) used in the original papers of FedAvg and FedMD. The initial grid $\{1.5, 1, 0.5, 0.1, 0.05, 0.01\}$ loosely covers good SGD learning rates and can be extended to scales such as $\{0.005, 0.001\}$ whenever necessary. A more fine-grained tuning only marginally improved the results of all methods and did not affect our conclusions. The learning rate decay used in appendix (i.e. decay by 10 at 50% and 75% of the local training epochs) follows the general scheme as in many published papers.

To distinguish the benefits of FedDF from the small learning rate or Adam optimizer, we report the results of using Adam (2e-3) for both local training and model fusion (over three seeds), on CIFAR-10 with ResNet-8 (setups in Fig. 2). For $\alpha = 1$, we have 80.27 v.s. 72.73 (local training via SGD) and 83.32 v.s. 78.13 (local training via Adam); for $\alpha = 0.1$, the results of FedDF v.s. FedAvg are 71.52 v.s. 62.44 and 72.58 v.s. 62.53 respectively. Improving the local training through Adam might help FL but the benefit vanishes with higher data heterogeneity (e.g. $\alpha = 0.1$). Performance gain from FedDF is robust to data heterogeneity and also orthogonal to effects of learning rates and Adam. As a side note, FedDF uses the common learning rate magnitude for Adam (different from SGD's 1e-1) and is not small.