Fed2KD: Heterogeneous Federated Learning for Pandemic Risk Assessment via Two-Way Knowledge Distillation

Chuanneng Sun, Tingcong Jiang, Saman Zonouz, and Dario Pompili
Department of Electrical and Computer Engineering, Rutgers University–New Brunswick, NJ, USA
Emails: {chuanneng.sun, tingcong.jiang, saman.zonouz, pompili}@rutgers.edu

***Abstract*—The world has suffered a lot from the COVID-19 pandemic. Though vaccines have been developed, we still need to be ready for its variants and other possible pandemics in the future. To provide people with pandemic risk assessments without violating privacy, a Federated Learning (FL) framework is envisioned. However, most existing FL frameworks can only work for homogeneous models, i.e., models with the same configuration, ignoring the preferences of the users and the various properties of their devices. To this end, we propose a novel two-way knowledge distillation-based FL framework, Fed2KD. The knowledge exchange between the global and local models is achieved by distilling the information into or out from a tiny model with unified configuration. Nonetheless, the distillation cannot be conducted without a common dataset. To solve this bottleneck, we leverage the Conditional Variational Autoencoder (CVAE) to generate data that will be used as a proxy dataset for distillation. The proposed framework is firstly evaluated on benchmark datasets (MNIST and FashionMNIST) to test its performance against existing models such as Federated Averaging (FedAvg). The performance of Fed2KD improves by up to 30% on MNIST dataset, and 18% on FashionMNIST when data is non-independent and identically distributed (non-IID) as compared to FedAvg. Then, Fed2KD is evaluated on the pandemic risk assessment tasks through a mobile APP we developed, namely DP4coRUna, which provides indoor risk prediction.**

***Index Terms*—Federated learning; Knowledge Distillation; COVID-19 Risk Assessment.**

I. Introduction

Overview: In the year of 2019, the Coronavirus disease 2019 (COVID-19) was discovered and has become a great threat to people's health all over the world. It takes several days from being infected by the virus for symptoms to show. In some cases, however, the symptoms could hide for up to 14 days, which is an extremely long period in which asymptomatic people can unknowingly infect relatives, friends, co-workers, and even anyone they get close to.

Motivation: There have been many platforms trying to help with the situation [1], [2]. However, these platforms have many limitations. They only report the number of infected in a region/country which can hardly help the individuals that trying to stay safe. It is well-known that COVID-19 is most dangerous when people are exposed to the virus in an enclosed space, which means being outside is far less dangerous than being inside a building or a room. These platforms apparently fail to tell which buildings or rooms are safe to the users. Besides these platforms, there are also several software trying to achieve *contact tracing* [3]. Contact tracing is an effective disease control measure and a key strategy for preventing further spread of the pandemic. Several app-based solutions are being developed to track the near-real-time spread of the COVID-19 disease and to identify which people are likely to be infected based on app users location history. However, most of the existing solutions have limitations, e.g., (i) they are reactive in nature, i.e., they detect the exposure after the close contact has happened; (ii) they adopt a centralized approach, which does not scale and is prone to single point of failure; (iii) they use Global Positioning System (GPS) for positioning, which does not work indoors; and (iv) they allow user-server/user-user data exchange which could lead to a violation to users' privacy. As such, a *proactive, scalable, privacy-preserving indoor* risk assessment framework is needed. Federated Learning (FL) suffices all of the requirements mentioned above and have proven its accurate, distributed and privacy-preserving nature in many applications [4], [5].

Contribution: To provide a solution to pandemic risk assessment, in this paper, we propose a two-way Knowledge Distillation (KD)-based FL framework, namely Fed2KD, which enables the training of *user-customized models* by leveraging a set of unified small models. We define two procedures during the training of the framework–forward distillation and backward distillation. During forward distillation, the complex models (i.e., user-configured) distill their knowledge about local data to the unified models. After forward distillation, unified models will be uploaded to the server and merged and then the clients will download the merged models for backward distillation, during which the knowledge about the global data distribution will be distilled into the local complex models. However, backward distillation becomes a bottleneck because only using the local data cannot distill the global knowledge out of the unified models and the user-configured models will learn nothing if we only use the local private data. To solve this bottleneck, we adopt the Conditional Variational Autoencoder (CVAE) algorithm to generate meaningful data from the global distribution without leaking any real data. Adding two models to the local clients may seem to be burdening in terms of memory and computation expenses, however, the sizes of them are very small comparing to the user-configured model. Furthermore, CVAEs only need to be trained for several epochs and then can be used for inference.

To evaluate Fed2KD and prove it outperforms existing FL frameworks, we first test it on benchmark datasets such as MNIST [6] and FashionMNIST [7] against existing frameworks such as FedAvg. Then we deploy it to the pandemic risk assessment task. We developed a Decentralized Proactive, Predictive, Personalized, Privacy-preserving (4P) COVID-19 recommendation App that: (1) is proactive in nature, which helps mitigate the spread of the virus significantly, e.g., it provides recommendations on which path is best to go from place A to B, which regions of the bus/train coach to sit, when to visit which stores in a mall, etc., thereby empowering the user by giving control on what to do, so as to minimize the risk of contagion while allowing us to continue to live our lives; (2) is endowed with built-in privacy and security features; (3) works both indoors and outdoors; (4) is a robust user-based distributed solution; (5) achieves high reliability by leveraging collaborative information fusion and model-based verification (more details will be discussed in Sect. IV).

The main contribution of this work are three fold:

1) We propose a two-way KD-based federated learning structure to train different types and sizes of models. The models learn from each other without getting access to any local private data.
2) We evaluate the framework on benchmark dataset such as MNIST and FashionMNIST that illustrate its high accuracy. We evaluate the model on different levels of Non-IID data to show that the model is able to deal with heterogeneous data distribution.
3) We evaluate the model on a phone APP that can provide risk prediction for COVID-19 pandemic. The results show that the model can work for such data type.

Paper Organization: The rest of the paper is organized as follows. In Sect. II, we position our work w.r.t. existing risk assessment frameworks. In Sect. III, we provide a brief background on federated learning and then present the details about our proposed Fed2KD. Finally, in Sect. IV, we evaluate the proposed framework on both a benchmark dataset and an APP we developed for COVID-19 risk assessment.

II. Related Work

Here, we first position our work w.r.t. existing frameworks for COVID-19 risk assessments and then we provide a brief introduction for Federated Learning (FL) and heterogeneous FL algorithms.

Risk Assessment: Chakraborty et al. [8] proposed a framework to provide real-time COVID-19 forecasting and risk assessments. For forecasting, they consider the COVID-19 infection data as time series data and use a hybrid of ARIMA and wavelet-analysis method for prediction. For risk assessment, a Regression Tree (RT) is applied and trained on a dataset of fatality ratio for 50 countries. The RT is trained using Mean Squared Error (MSE) as criteria for splitting and a pruning technique to get rid of the redundancy. They can provide relatively accurate predictions for risks and yet the granularity of the model is too coarse and people cannot use it for their own safety.

Misra et al. [9] proposed a Q Learning-based COVID-19 navigation tool for avoiding high risk areas and they minimized the training and response time of the service by leveraging the fog computing architecture. During the training of the Q learning agent, they introduce a penalty so that the agent can learn to recommend routes with zero/lower risks. However, their method only works for outdoor environment, which indicates that the help their method can provide is limited because people have the greatest chance to get COVID-19 when exposed to it in an indoor environment.

Federated Learning: Besides the introduction for existing risk assessment frameworks, we will also provide a brief review for federated learning algorithms.

McMahan et al. [4] proposed FedAvg, which opened the gate to FL algorithms. FedAvg is an algorithm that can jointly train distributed models without getting access to their data by iteratively merging the model parameters. Nonetheless, FedAvg only considers homogeneous models and did not incorporate user-customizability.

Wang et al. [10] proposed to use double Q learning on the client selection phase of federated learning. The define the state space for the Q learning agent as the decomposed parameters of the client models by Principal Component Analysis (PCA). It can be considered as an intelligent clustering mechanism assuming that there is an optimal policy that can identify the best group of clients to be merged. However, the proposed work can only work for models with identical configuration and it cannot provide a solution to heterogeneous models.

Li et al. [11] proposed a framework that works for heterogeneous models, namely FedMD. A public dataset is introduced and the local models will generate outputs using this dataset. The server will calculate the mean of the outputs collected from the clients and use it as a consensus which is used as a target for the clients to meet. In this way, the local models are trained towards the same direction without leaking any data. However, this framework requires a public dataset which could introduce potential risks to users' privacy.

Zhu et al. [12] proposed a generator-based federated learning scheme where each client trains a generator and at each time step the generator will be uploaded to the server and merged by applying a weighted average. The local models are trained on the local data it possesses and the data generated by the generator. This method transforms federated learning problem into federated data generation problem, however, the data generated by the generator could lead to a unstable convergence for the classifiers.

Chen et al. [13] proposed FedDistill, which is a federated learning framework with distributed Bayesian models to form an ensemble model. It is assumed that unlabeled data can be collected by the centralized server and this dataset will be used as the proxy dataset to train the local models to approach the global ensemble model.

III. Proposed Work

We now firstly review the background for federated learning and on knowledge distillation in Sect. III-A. Then we introduce our proposed Fed2KD framework in Sect. III-B.

A. Background

Federated Learning: Federated learning is a framework aiming to optimize a global model by iteratively training and aggregating local models trained on local data [4]. Suppose the FL framework has been set up for a C-class classification problem and the feature space and label space are defined as $\mathcal{X}$ and $\mathcal{Y} = [C]$, respectively. Let $\mathcal{D} = \{(x_1, y_1), ..., (x_M, y_M)\}$ represent the dataset where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ are the features and labels and M is the size of the dataset. The classifier (e.g., neural network and support vector machine) is denoted as $f : \mathcal{X} \to \mathcal{S}$ with $\mathcal{S} = \{\boldsymbol{z} | \sum_{i=1}^{C} z_i = 1, z_i \geq 0, \forall i \in \mathcal{Y}\}$ as the probability space. Vector $\boldsymbol{z} \in \mathbb{R}^C$ represents a probability distribution over C classes.

Assuming that Cross Entropy (CE) loss is used, the classification problem can be written as,

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{C} p(y = i)\mathbb{E}_{x|y=i}\left[log f(x; \boldsymbol{\theta})\right], \tag{1}$$

where $\boldsymbol{\theta}$ denotes the parameters of the classifier f

In FL, (1) becomes different with distributed models and unsharable data. Suppose we have K clients participating in the FL training procedure, each client has its own dataset $\mathcal{D}^k = \{(x_i^k, y_i^k)\}$ with distribution $p^{(k)}$. If $\boldsymbol{\theta}_t$ denotes the merged weights downloaded from the server at time step t and $\boldsymbol{\theta}_t^{(k)}$ represents the local weights after local optimization at client k, the weight update for local model is,

$$\boldsymbol{\theta}_t^{(k)} = \boldsymbol{\theta}_{t-1}^{(k)} - \eta \sum_{i=1}^{C} p^{(k)}(y = i) \nabla_{\boldsymbol{\theta}} \mathbb{E}_{x|y=i}\left[log f(x; \boldsymbol{\theta}_{t-1}^{(k)})\right], \tag{2}$$

where η is the factor for controlling the step size (a.k.a. learning rate). After the local update, the weights of the model are uploaded to the server for merging. Let $M^{(k)}$ denote the size of the dataset on client k, the merge can be written as,

$$\boldsymbol{\theta}_{t+1} = \sum_{k=1}^{K} \frac{M^{(k)}}{\sum_{k=1}^{K} M^{(k)}} \boldsymbol{\theta}_{t+1}^{(k)}, \tag{3}$$

which can be considered as an average of the local parameters weighted by the fraction of data the client has w.r.t. the entire dataset.

Knowledge Distillation: Knowledge Distillation (KD) is also known as the teacher-student framework which aims at learning a light-weight student model by distilling the knowledge from the teacher [14], [15]. KD frameworks usually involve a proxy dataset, $\mathcal{D}^p$, on which the teacher and the student are trained for knowledge extraction. The outputs of the teacher and student are collected to calculate the distance as the objective to be optimized. Kullback-Leibler divergence (KL divergence) is a popular metric to measure the distance of

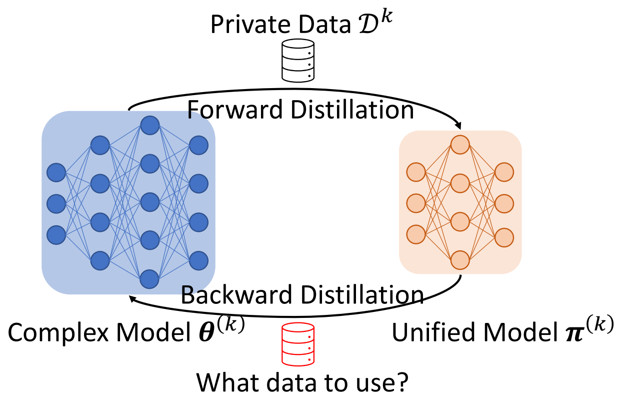

Private Data $\mathcal{D}^k$
Forward Distillation
Backward Distillation
Complex Model $\boldsymbol{\theta}^{(k)}$
Unified Model $\boldsymbol{\pi}^{(k)}$
What data to use?

Fig. 1: Illustration of forward and backward knowledge distillation.

the distributions given by the teacher and student. Let $\boldsymbol{\theta}^{(T)}$ and $\boldsymbol{\theta}^{(S)}$ denote the parameters for the teacher model and student model, respectively. The objective of KD can be written as,

$$l(\boldsymbol{\theta}^{(S)}) = D_{KL}\left[\sigma(f(x; \boldsymbol{\theta}^{(T)})) || \sigma(f(x; \boldsymbol{\theta}^{(S)}))\right] \tag{4}$$

$$= \mathbb{E}_{x,y \sim \mathcal{D}^p}\left[log \frac{\sigma(f(x; \boldsymbol{\theta}^{(T)}))}{\sigma(f(x; \boldsymbol{\theta}^{(S)}))}\right], \tag{5}$$

where $\sigma(\cdot)$ is the non-linear activation function and D_{KL} is the KL divergence. By back-propagating the KL divergence loss between the outputs from two models, the student model can learn how much difference between its output and the output from the teacher model, and thus it can learn to behave like the teacher.

B. Two-way Knowledge Distillation for Federated Learning

As mentioned in Sect. I, we consider heterogeneous user models, which indicates that the traditional train-and-average method for merging models in FedAvg is not working in this setting. To formally define the proposed framework, we adopt the notation from Sect. III-A. With distributed models and data, FL is updated by extracting the local knowledge from local models and merging them to get the global knowledge. However, (3) does not work for heterogeneous models because different sizes of parameters (i.e., matrices) cannot be summed and averaged.

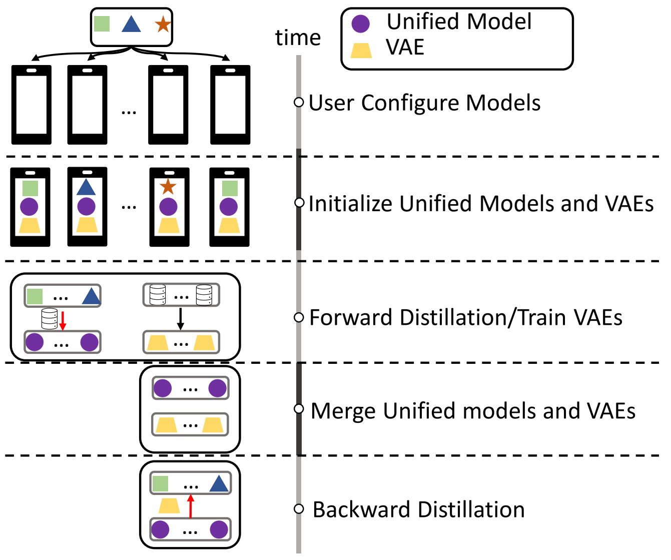
We solve this problem by introducing knowledge distillation to the FL framework. The intuition is that we distill the knowledge out of the local models and merge them to get the average and then send them back to the clients. Nonetheless, the existing KD frameworks are mostly one-to-one and many-to-one and applying these frameworks will introduce significant computation and communication overhead. For example, if we adopt the pairwise KD method, we have to run the distillation procedure $\binom{N}{2}$ times.

Our solution to this problem is to leverage a set of small models parameterized by $\boldsymbol{\pi}^{(k)}$ with the unified configurations deployed on the clients which will help the heterogeneous models to converge to the consensus. The complex models (i.e., the original local heterogeneous models) and the unified models will form teacher-student relationships in turn in KD paradigm. The complex model will first be trained on the local dataset and then transfer the knowledge to the unified model. The unified model will then be uploaded to the server for aggregation following (3). After merging, the unified models are supposed to contain the global knowledge and will be downloaded to the clients. Then, the small models will distill their knowledge to the complex models so that the big ones are updated with new knowledge from others, that is, the complex model and the small model form a knowledge transfer circle. Adopting (4), the distillation from complex models to unified models can be written as,

$$\min_{\boldsymbol{\pi}^{(k)}} \mathbb{E}_{x,y\sim\mathcal{D}^k} \left[D_{KL} \left[\sigma(f(x;\boldsymbol{\theta}^{(k)})) || \sigma(f(x;\boldsymbol{\pi}^{(k)})) \right] \right]. \quad (6)$$

However, there is a problem with this setting. The forward distillation (i.e. distill knowledge from complex models to small models) can be easily achieved by training the models on the local dataset with cross-entropy loss and KL divergence loss and yet the backward distillation seems to be impossible because only local data can be used which will not trigger the small models' knowledge about the others. Fig. 1 provides a more illustrative example. As is shown, we can distill the knowledge of the complex model using the local data because the knowledge it has is local, but we cannot distill the knowledge that unified models have because the knowledge is global and the data that is available is local. Thus, in order to solve this problem, we need to find a way to distill the knowledge out of the small models. The most straightforward way is to share data across clients as in Zhao et al. work [16] and the results show that by sharing a small fraction of data, the performance of models can be greatly improved. However, this violates the hard bottomline of FL which is no data sharing. Therefore, a data generation method that is reliable and do not require data sharing is what we are looking for. The most ideal method is to generate a dataset of infinite size with every possible value combination in the feature spaces so that the complex model can learn to mimic the small model in every possible way until they contain the same knowledge. However, this is quite absurd due to memory and time constraints. One more realistic way is to generate meaningful noise from the distribution of the dataset. Bearing that in mind, we introduce Conditional Variational Autoencoder (CVAE).

Conditional Variational Autoencoder: A popular method for generating data is Variational AutoEncoder (VAE) [17]. VAE cosists of an encoder, which maps the original data to the latent space, and a decoder, which maps the data in the latent space back to the original form. Besides the encoder-decoder pair, VAE also introduces a latent variable into the framework. Formally speaking, the autoencoder assumes that the original data x is conditioned on a latent variable z. Based on the scheme described, VAE regards the latent variable as a random variable and the encoding and decoding processes become probabilistic processes. With that being said, we can represent the encoder and decoder as $p(z|x)$ and $p(x|z)$, respectively. Assuming that $p(x|z)$ is Gaussian with a mean defined by



time
● Unified Model
VAE
...
User Configure Models
Initialize Unified Models and VAEs
Forward Distillation/Train VAEs
Merge Unified models and VAEs
Backward Distillation

Fig. 2: The workflow of the Fed2KD framework. Red arrows indicate the knowledge distillation direction.

a function of latent variable z and a covariance matrix with form cI, where c is a constant and I is the identity matrix, the decoder can be further written as $p(x|z) = \mathcal{N}(f(z), cI)$. As for the encoder, VAE tries to approximate it using a Gaussian distribution $q_x(z)$ with a mean defined by $g(x)$ and variance defined by $h(x)$. The encoder can now be represented as $p(z|x) = q_x(z) = \mathcal{N}(g(x), h(x))$. With the definitions above, the encoding process becomes generating the mean and variance for a Gaussian distribution and the decoding becomes sampling z from the distribution and decode it. Moreover, to enable the gradients to be propagated through the sampling process, an important technique, the reparameterization trick, is introduced. The sampling process now becomes the addition of a random noise and a deterministic process through which the gradients can be passed. With the trick, the sampling of random variable z can be written as,

$$z = h(x)\zeta + g(x) \quad (7)$$

where $\zeta \sim \mathcal{N}(0, I)$ is random Gaussian noise. With a trained VAE, we can just feed random noise to the decoder and it will output data that approximate the original dataset. In our proposed work, we use a variation of the VAE, namely Conditional VAE, which conditions on the label information so that the generator can generate data according to labels instead of just random noise.

With the help of CVAE, we can formulate the backward distillation as,

$$\min_{\boldsymbol{\theta}^{(k)}} \mathbb{E}_{\hat{x}\sim G(\cdot;\hat{y},\boldsymbol{\eta}^{(k)}),\hat{y}\sim p(y)} \left[D_{KL} \left[\sigma(f(\hat{x};\boldsymbol{\pi}^{(k)})) || \sigma(f(\hat{x};\boldsymbol{\theta}^{(k)})) \right] \right], \quad (8)$$

where $G(\cdot;\hat{y},\boldsymbol{\eta}^{(k)})$ denotes the CVAE on the ith client parameterized by $\boldsymbol{\eta}^{(k)}$ and $\hat{x}$ and $\hat{y}$ are data generated by CVAE, respectively.

Algorithm 1 Fed2KD Algorithm

```
1: User configure $\boldsymbol{\theta}_0^{(1)},...,\boldsymbol{\theta}_0^{(K)}$
2: Initialize Uni Model $\boldsymbol{\pi}_0$
3: Initialize CVAE Model $\boldsymbol{\eta}_0$
4: for round $t = 1, 2, ...$ do
5:     for each client $k \in K$ in parallel do
6:         $\boldsymbol{\pi}_{t+1}^{(k)}, \boldsymbol{\eta}_{t+1}^{(k)} \leftarrow ForwardDistill\left(\boldsymbol{\theta}_t^{(k)}, \boldsymbol{\eta}_t^{(k)}\right)$
7:     $\boldsymbol{\pi}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{M^{(k)}}{\sum_k M^{(k)}} \boldsymbol{\pi}_t^{(k)}$
8:     $\boldsymbol{\eta}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{M^{(k)}}{\sum_k M^{(k)}} \boldsymbol{\eta}_t^{(k)}$
9:     for each client $k \in K$ in parallel do
10:        $\boldsymbol{\theta}_{t+1}^{(k)} \leftarrow BackwardDistill(\boldsymbol{\pi}_{t+1}^{(k)}, \boldsymbol{\eta}_{t+1}^{(k)})$
11: procedure FORWARDDISTILL($\boldsymbol{\theta}_t^{(k)}, \boldsymbol{\eta}_t^{(k)}$)
12:     $\boldsymbol{\theta}_t^{(k)} \leftarrow$ Update $\boldsymbol{\theta}_t^{(k)}$ according to Eq. (2)
13:     $\boldsymbol{\pi}_{t+1}^{(k)} \leftarrow$ Update $\boldsymbol{\pi}_t^{(k)}$ according to Eq. (6)
14:     $\boldsymbol{\eta}_{t+1}^{(k)} \leftarrow$ Update $\boldsymbol{\eta}_t^{(k)}$ on local data $\mathcal{D}^k$
15:     Return $\boldsymbol{\pi}_{t+1}^{(k)}, \boldsymbol{\eta}_{t+1}^{(k)}$
16: procedure BACKWARDDISTILL($\boldsymbol{\pi}_{t+1}^{(k)}, \boldsymbol{\eta}_{t+1}^{(k)}$)
17:     Generate $\hat{y} \sim p(y)$
18:     Generate $\hat{x} \sim G(\cdot; \hat{y}, \boldsymbol{\eta}_{t+1}^{(k)})$
19:     $\boldsymbol{\theta}_{t+1}^{(k)} \leftarrow$ Update $\boldsymbol{\theta}_t^{(k)}$ according to Eq. (8)
20:     Return $\boldsymbol{\theta}_{t+1}^{(k)}$
```

Now we have every part for our framework, but we have to stop and think of a question: why don't we just use CVAEs trained in a federated manner to help the complex models? We answer this question by presenting our results in Sect. IV-A and claim that our method has a better performance than directly using CVAEs.

The whole workflow of Fed2KD is shown in Fig. 2 and the detailed diagram can be found in Algorithm 1. The framework is initiated by user configuring the client models and initializing the unified models and CVAEs. Then, we will train the complex models and CVAEs using local data. The trained complex models will distill its knowledge to the unified small models which will be uploaded and merged after the forward distillation. After model aggregation on the server's side, the clients will download the unified models and CVAEs and perform the backward distillation using the data generated by CVAE. The framework keep iterating between forward and backward distillation until a stop criteria is met.

IV. Evaluation

To evaluate the proposed algorithm, we first test it on the benchmark datasets to show that the performance of the model is accurate by comparing it with existing methods in Sect. IV-A. Then, in Sect. IV-B, we evaluate the algorithm on real risk prediction task through an APP we developed, named DP4coRUna. The algorithm framework is embedded in the APP and We took advantage of the AWS server to form a centralized learning architecture.

A. Benchmark Dataset Evaluation

Experiment Setup: We use 10 clients for the experiments and each of them has a configurable complex model, a small model and a VAE model. The complex models can be MLP or CNN and the sizes can be anything that the device can afford. We use five Multilayer Perceptrons (MLPs) of size [128, 256, 256] and five CNNs with three convolution modules where each module contains a convolution layer and a max-pooling layer, in our experiment. We use a small MLP of size [256] as the unified model in our experiment. The VAE model is also realized in a MLP form of size [256] as encoder and [256] as decoder. All models are using learning rate = 0.01 with Adam [18] optimizer. The framework and the experiment are implemented in Python using PyTorch framework [19]. We set local update epoch as 1 which means that we only perform one epoch of local update on the private data and then conduct the forward and backward distillation. Each client possesses 500 samples of data and the batch size is 64.

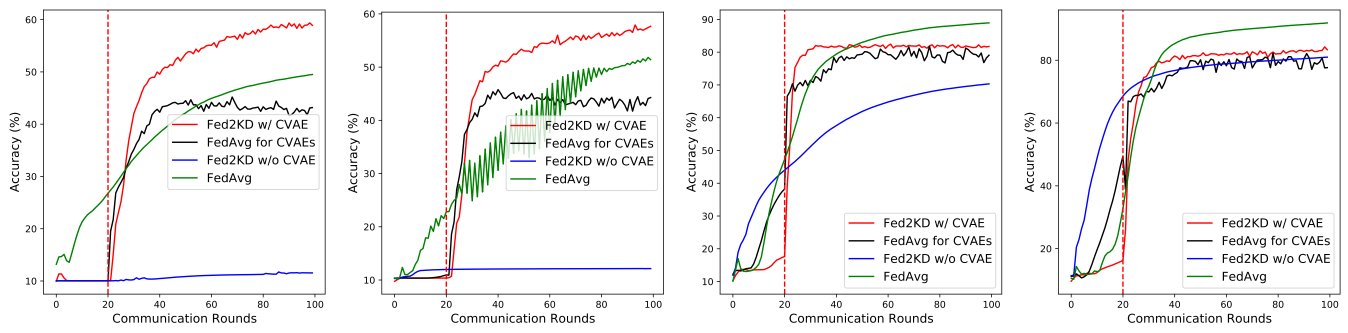
We compare the proposed Fed2KD with three other models– (i) Fed2KD without CVAE, in which only complex models and unified models are deployed on each client using only the local data for both forward and backward distillation, (ii) FedAvg for CVAE, in which only complex models and CVAE are used and the whole framework becomes how to train a good generator using FedAvg on CVAE models, and (iii) FedAvg, which is used as baseline. However, FedAvg only works for model with the same configuration and, to compare it with other heterogeneous frameworks, we train two FedAvg frameworks for each dataset where one uses 10 MLPs of size [128, 256, 256] and the other uses 10 CNNs with three convolution modules.

Dataset: To test how well the model performs, we run it on two benchmark datasets, MNIST [6] and FashionMNIST [7]. MNIST is a dataset for handwritten digits recognition task and it consists of 70,000 images of digits from 0 to 9 of size 28×28, which are divided into 10,000 testing set and 60,000 training set. Similar to MNIST, FashionMNIST dataset is a dataset of Zalando's article images, including pictures of shoes, trousers and so on, and it also contains 60,000 examples for training set and a test set of 10,000 examples.

Data Heterogeneity: We follow the data generation method from Hsu et al. [20], in which the data are drawn from a Dirichlet distribution $Dir(\alpha)$ with parameter $\alpha \in (0, \infty)$ to control the data heterogeneity. The greater α, the less concentrated the data distribute over the clients, i.e., with $\alpha \to \infty$, the data will be identically distributed over the clients and as $\alpha \to 0$, each client will only possess the data from one class. For our experiments, we choose four values for α.

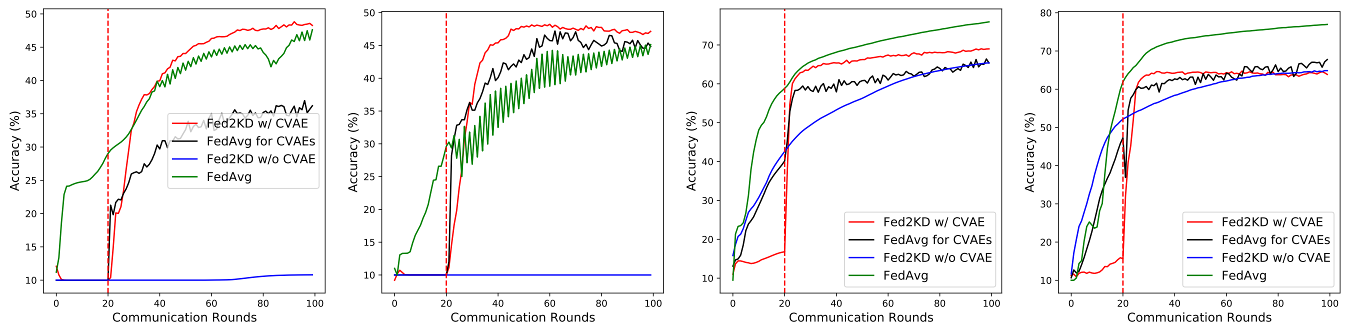
Results: Figs. 3, 4, and 5 present the results of our experiments on MNIST and FashionMNIST datasets, respectively. More detailed results can be found in Tables I and II. We can observe several interesting facts from the results:

1) When α is high, all four frameworks work well. The reason to this outcome is that the difference of client data distribution is not very significant and the model



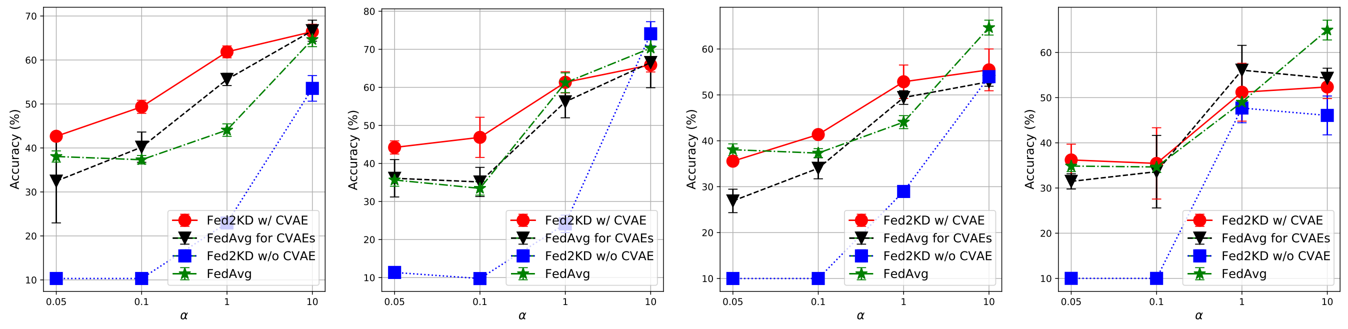
(a) (b) (c) (d)

Fig. 3: The testing accuracy of (a) MLPs and (b) CNNs trained on MNIST dataset with $\alpha = 0.05$ and the testing accuracy of (c) MLPs and (d) CNNs trained on MNIST dataset with $\alpha = 10$. The red dashed vertical line at x=20 indicates the start of using CVAE.



(a) (b) (c) (d)

Fig. 4: The testing accuracy of (a) MLPs and (b) CNNs trained on FashionMNIST dataset with $\alpha = 0.05$ and the testing accuracy of (c) MLPs and (d) CNNs trained on FashionMNIST dataset with $\alpha = 10$.



(a) (b) (c) (d)

Fig. 5: The testing accuracy of (a) MLPs and (b) CNNs trained on MNIST dataset and (c) MLPs and (d) CNNs trained on FashionMNIST dataset with different α values.

can learn a good global knowledge even on its local data. As α decreases, the performance of the framework without CVAE drops significantly. This is because the distances between client data distributions are getting larger and simply training on local data cannot learn any global knowledge.

2) When α approaches 0.05, the proposed Fed2KD outperforms FedAvg for CVAEs. This is because that the CVAE models does not converge to a stable state and it cannot provide trustworthy samples to the complex models for the backward distillation.
3) The red dashed line at $x = 20$ indicates that we start to let CVAE to be involved in the training process. It is obvious that with the help of CVAE, both Fed2KD and Fed2KD without unified models are improved significantly. Thus, we could claim that with data generated by CVAE, the bottleneck of backward distillation is eased.

In Fig. 5, we observe that for both datasets, when α increases, the model performance improves. Fed2KD w/o CVAE is the most improved model, which is because the backward distillation is completely blocked when α is small; as the data are distributed more evenly, it gets access to more data in other classes and, thus, the performance increases. Furthermore, FedAvg achieves better performance when α has a greater value, which is within expectation because FedAvg can handle well IID-distributed data.

B. *Mobile Risk Assessment*

The evaluation of Fed2KD on benchmark datasets show that it is a good and reliable framework for federated learning.

TABLE I: Test accuracy for models trained on MNIST dataset.

Test Accuracy on MNIST (%)			
Model	α	**MLP**	**CNN**
Fed2KD	0.05	58.90 ± 1.84	57.68 ± 0.36
FedAvg for CVAE		43.17 ± 5.93	44.27 ± 5.44
Fed2KD w/o CVAE		11.54 ± 0.03	12.14 ± 0.06
FedAvg		37.91 ± 1.28	35.65 ± 1.65
Fed2KD	0.1	65.72 ± 2.14	66.59 ± 2.11
FedAvg for CVAE		52.90 ± 9.30	51.06 ± 9.56
Fed2KD w/o CVAE		13.79 ± 0.03	15.83 ± 0.02
FedAvg		35.21 ± 1.29	33.45 ± 1.78
Fed2KD	1	67.70 ± 0.49	59.78 ± 5.75
FedAvg for CVAE		60.68 ± 1.72	63.24 ± 5.90
Fed2KD w/o CVAE		34.51 ± 1.73	37.40 ± 0.21
FedAvg		50.75 ± 1.99	61.20 ± 2.61
Fed2KD	10	68.99 ± 2.67	63.87 ± 0.98
FedAvg for CVAE		65.36 ± 1.22	67.76 ± 4.05
Fed2KD w/o CVAE		65.40 ± 0.67	64.87 ± 0.13
FedAvg		69.39 ± 2.95	70.30 ± 3.46

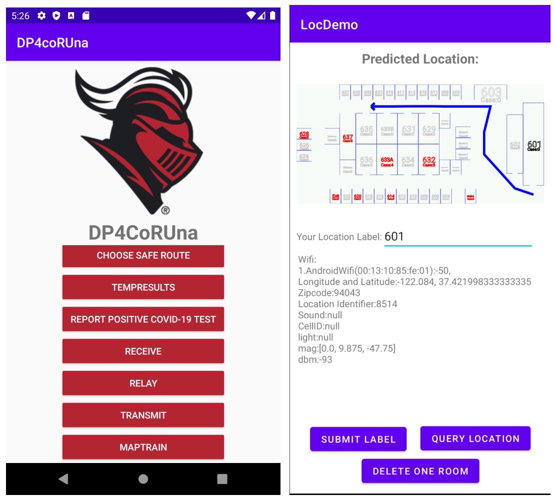
TABLE II: Test accuracy for models trained on FashionMNIST dataset.

Test Accuracy on FashionMNIST (%)			
Model	α	**MLP**	**CNN**
Fed2KD	0.05	48.27 ± 4.14	45.00 ± 0.75
FedAvg for CVAE		36.20 ± 3.21	38.52 ± 2.97
Fed2KD w/o CVAE		10.80 ± 0.02	12.00 ± 0.03
FedAvg		37.74 ± 1.24	34.86 ± 1.23
Fed2KD	0.1	55.71 ± 1.29	47.15 ± 11.13
FedAvg for CVAE		44.39 ± 2.89	44.84 ± 5.36
Fed2KD w/o CVAE		15.30 ± 0.05	$9.99 \pm 0.$
FedAvg		37.34 ± 0.97	34.66 ± 1.22
Fed2KD	1	67.70 ± 0.49	59.78 ± 5.75
FedAvg for CVAE		60.68 ± 1.72	63.24 ± 5.90
Fed2KD w/o CVAE		34.51 ± 1.73	37.40 ± 0.21
FedAvg		44.06 ± 1.42	48.99 ± 1.58
Fed2KD	10	68.99 ± 2.67	63.87 ± 0.98
FedAvg for CVAE		65.36 ± 1.22	67.76 ± 4.05
Fed2KD w/o CVAE		65.40 ± 0.67	64.87 ± 0.13
FedAvg		64.65 ± 1.62	64.95 ± 2.19

Therefore, in this section, we introduce a mobile APP we developed which utilizes Fed2KD for pandemic risk assessment. The APP has three core modules:

(i) *Localize+Mark* provides accurate indoor user localization and k-anonymity-based infection marking. Localize+Mark first obtains the route information (source and destination) directly from the user. It then calculates and marks the infectious risk associated with each of these candidate paths/activities using the indoor/outdoor location information of users associated with such paths/activities using k-anonymity mechanism that preserves user privacy. The indoor/outdoor localization functionality only needs Wi-Fi Access Points (AP) as input (i.e., features) and it can provide accurate prediction for users locations. Data collected from different locations are tagged with different labels with the associated features (e.g., Wi-Fi APs and signal strengths). In this way, a distributed database is built and, to to query a user's location, we only need to search the database for a match of the features.

(ii) *ContTrace* determines people who have been there recently, where the database entries of two devices are compared to find a similarity match. ContTrace identifies the people who had been in contact with a particular person or location (say 'specimen') in the past two weeks, which is accomplished



Your Location Label: 601

Wifi:
1.AndroidWifi(00:13:10:85:fe:01):-50,
Longitude and Latitude:-122.084, 37.421998333333335
Zipcode:94043
Location Identifier:8514
Sound:null
CellID:null
light:null
mag:[0.0, 9.875, -47.75]
dbm:-93

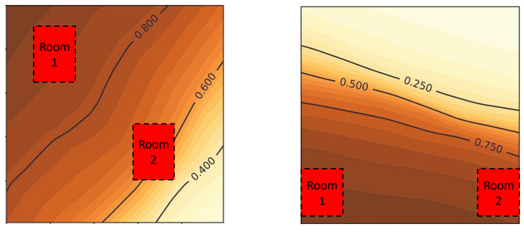
(a) (b)
Fig. 6: Screenshot demos of DP4coRUna.

as follows. The database records (i.e., the locations and their sensor signatures) of specimen corresponding to the last two weeks are queried over the ToR network to identify devices whose databases have matching information in the last two weeks. When a device receives these query records, it initiates a comparison check to see if any of the query records are similar to the records in its database. If it is so, then the app alerts the user that s/he had been in potential contact with an infected person in the past two weeks and suggests the user to self-quarantine and contact the health provider. When this happens the app also presents the matching records (with sensor signatures including Wi-Fi, Cell ID, etc.) in its database that are similar to the query for user validation. We note that this software module can be replaced with other solutions, e.g., Apple/Google app if needed, while still reaping the benefits of the other modules.

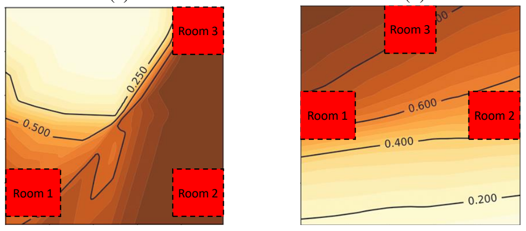
(iii) *Recommend*, the last and final module, then takes the outputs of the previous modules to generate the top recommendations for the user in decreasing order of strength. The user can then choose the option that best fits his/her requirements. In this way, our app provides the users with valuable information and puts the users in a position to make the best decision for themselves.

Demo of Fed2KD with DP4coRUna: To illustrate how to apply Fed2KD on COVID-19 scenario, we start by introducing the data flow of the APP. After downloading the APP, the user needs to firstly report whether he/she is positive. Fig. 6(a) shows the User Interfaces (UIs) for the front page and the entrance to report positive. After reporting, the APP will train the Fed2KD model on the mobile devices of users who reported positive. The training goes on for several iterations and a global risk map will be generated from the training process. With this map, the top k safest can be found which will be provided to users to choose.

We showcase an indoor example in Fig. 7 assuming that there are n users and each of them has been to one of the n rooms with no replacement, which means that each device has the data of one unique dangerous zone. The high-risk



(a) (b)



(c) (d)

Fig. 7: Heatmaps for risks scores generated in an indoor floor; the rooms in red represent the ground truth labels, while surrounding heatmaps are the risk scores from Fed2KD.

rooms have been marked in red color in Figs. 7. With the help of *Localize+Mark* and *ContTrace* modules, the clients' devices are able to know where the users have been and thus generate data and labels correspondingly. For example, user A that is tested positive has been to room A, then data $\{(LocationOfRoomA, 1)\}$ will be generated where 1 stands for positive and 0 represents negative. With multiple users, a federated learning structure is formed and, after following the training procedure, when another user queries the server for a risk over the whole area, he/she will get results as shown in Figs. 7. We can observe that the risk scores in the locations of the high-risk zones are significantly higher than other areas and the margins have been softened to consider the spread of viruses. With the risk scores, the APP can calculate top k safest routes, where k is a predefined number, from a starting location to a destination with the *Recommend* module. Fig. 6(b) shows an example routing demonstration using the 6th floor of Rutgers CoRE building. The risky rooms are marked as red and the user is trying to travel from room 601 to room 635. The recommended route in blue color directs the user to avoid the dangerous rooms.

V. Conclusion

We presented Fed2KD, a two-way knowledge distillation-based federated learning framework that can jointly train heterogeneous models of different types and sizes. We defined the forward and backward distillations during its training, and identified the backward distillation bottleneck. To solve this bottleneck, we leveraged the Conditional Variational Autoencoder (CVAE) to generate the proxy dataset for backward distillation where the unified models distill their global knowledge to the complex models. To evaluate the proposed model, we first tested it on two benchmark datasets (MNIST and FashionMNIST) and compared it with FedAvg for CVAE and Fed2KD without CVAE. The results showed that the performance of the proposed model is better than for the other two methods. Then, we evaluated the model on our COVID-19 risk assessment mobile APP. We assumed multiple rooms in a building infected by test-positive individuals and generated risk scores for the whole floor map. The region around the rooms showed a significant higher score than other areas.

Acknowledgement: This work was supported by the NSF RTML Award No. CCF-1937403.

References

[1] Mistletoer: China-covid-19. [Online]. Available: https://github.com/Minghou-Lei/COVID-19-historical-data-visualization-2019-nCoV-

[2] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track covid-19 in real time," *The Lancet infectious diseases*, vol. 20, no. 5, pp. 533–534, 2020.

[3] N. H. Repsonse. (2020) Covid trace. [Online]. Available: https://nvhealthresponse.nv.gov/covidtrace/

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[5] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[6] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[7] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[8] T. Chakraborty and I. Ghosh, "Real-time forecasts and risk assessment of novel coronavirus (covid-19) cases: A data-driven analysis," *Chaos, Solitons & Fractals*, vol. 135, p. 109850, 2020.

[9] S. Misra, P. K. Deb, N. Koppala, A. Mukherjee, and S. Mao, "S-nav: Safety-aware iot navigation tool for avoiding covid-19 hotspots," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6975–6982, 2020.

[10] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[11] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[12] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," *arXiv preprint arXiv:2105.10056*, 2021.

[13] H.-Y. Chen and W.-L. Chao, "Feddistill: Making bayesian model ensemble applicable to federated learning," *arXiv e-prints*, pp. arXiv–2009, 2020.

[14] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[20] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.