Federated Learning of Models Pre-Trained on Different Features with Consensus Graphs

Tengfei Ma[1] **Trong Nghia Hoang**[2] **Jie Chen**[3,1]

[1]IBM Research
[2]Washington State University
[3]MIT-IBM Watson AI Lab

Abstract

Learning an effective global model on private and decentralized datasets has become an increasingly important challenge of machine learning when applied in practice. Existing distributed learning paradigms, such as Federated Learning, enable this via model aggregation which enforces a strong form of modeling homogeneity and synchronicity across clients. This is however not suitable to many practical scenarios. For example, in distributed sensing, heterogeneous sensors reading data from different views of the same phenomenon would need to use different models for different data modalities. Local learning therefore happens in isolation but inference requires merging the local models to achieve consensus. To enable consensus among local models, we propose a feature fusion approach that extracts local representations from local models and incorporates them into a global representation that improves the prediction performance. Achieving this requires addressing two non-trivial problems. First, we need to learn an alignment between similar feature components which are arbitrarily arranged across clients to enable representation aggregation. Second, we need to learn a consensus graph that captures the high-order interactions between local feature spaces and how to combine them to achieve a better prediction. This paper presents solutions to these problems and demonstrates them in real-world applications on time series data such as power grids and traffic networks.

1 INTRODUCTION

To improve the scalability and practicality of machine learning applications in situations where training data are becoming increasingly decentralized and proprietary, Federated Learning (FL) [McMahan et al., 2017, Yang et al., 2019a, Li et al., 2019, Kairouz et al., 2019] has been proposed as a new model training paradigm that allows data owners to collaboratively train a common model without having to share their private data with others. The FL formalism is therefore poised to resolve the computation bottleneck of model training on a single machine and the risk of privacy violation, in light of recent policies such as the General Data Protection Regulation [Albrecht, 2016].

However, FL requires a strong form of homogeneity and synchronicity among the data owners (clients) that might not be ideal in practice. First, it requires all clients to agree in advance to a common model architecture and parameterization. Second, it requires clients to synchronously communicate their model updates to a common server, which assembles the local updates into a global learning feedback. This is rather restrictive in cases where different clients draw observations from different data modalities of the phenomenon being modeled. It leads to heterogeneous data complexities across clients, which in turn requires customized forms of modeling. Otherwise, enforcing a common model with high complexity might not be affordable to clients with low compute capacity; and vice versa, switching to a model with low complexity might result in the failure to unlock important inferential insights from data modalities.

A variant of FL [Hardy et al., 2017, Hu et al., 2019, Chen et al., 2020], named vertical FL (VFL), has been proposed to address the first challenge, which embraces the concept of vertically partitioned data. This concept is figuratively named through cutting the data matrix vertically along the feature axis, rather than the data axis. Existing approaches maintain separate local model parameters distributed across clients and global parameters on a central server. All parameters are then learned together, which causes a practical drawback:

Coordination overhead among clients and the central server, such as engineering protocols that enable multi-

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

ple rounds of communication (i.e., synchronicity) and coordination effort (i.e., homogeneity) to converge on universal choices of models and training algorithms, would be required, which can be practically expensive depending on the scale of the application.

To mitigate both constraints on homogeneity and synchronicity[1] satisfactorily, we ask the following question and subsequently develop an answer to it:

Can we separate global consensus prediction from local model training?

As shown later in our experiments, we will address this question in a real-world context of the national electricity grid, over which thousands of phasor measurement units (PMUs) were deployed to monitor the grid condition and data were recorded in real-time by each PMU [Smartgrid.gov]. PMU measurements, as time series data, are owned by several parties, each of which may employ different technologies leading to heterogeneous recordings under varying sampling frequencies and measured attributes. These data can be used to train machine learning models that identify grid events (e.g., fault, oscillation, and generator trip). Such an event detection system relies on collective series measurements at the same time window but distributed across owners. Using VFL to build a common model on such decentralized and heterogeneous data is plausible but not practical, because of a lack of autonomy to facilitate coordination among owners.

Main Contribution. To resolve the challenge, we introduce a feature fusion perspective to this setting, which aims to minimize coordination among clients and maximize their autonomy via a local–global model framework. Therein, each client trains a customized local model with its data modalities. The training is independent and incurs no coordination. Once trained, local feature representations of each client can then be extracted from the penultimate layer of the corresponding local models. Then, a central server collects and aggregates these representations into a more holistic global representation, used to train a model for global inference. There are two technical challenges that need to be addressed to substantiate the envisioned framework.

C1. There is an ambiguity regarding the correspondence between components of local feature representations across different clients. This ambiguity arises because local models were trained separately in isolation and there is no mechanism to enforce that their induced feature dimensions would be aligned. As a matter of fact, it is possible to permute the induced feature dimensions without changing the prediction outcome. Thus, if two models are trained separately, they might end up looking at the same feature space but with permuted dimensions.

[1]Note that in our case, synchronicity requires co-training among clients which is a weaker constraint than its usual meaning of further requiring clients to synchronize their updates per iteration.

C2. There are innate local interactions among subsets of clients that need to be accounted for. Naively concatenating or averaging the local feature representations accounts for the global interaction but ignores such local interactions, which are important to boost the accuracy of global prediction as shown later in our experiments.

To address **C1**, note that the feature dimension alignment problem is discrete in nature; furthermore, there is no direct feedback to optimize for such alignment. To sidestep this challenge, we develop a neuralized alignment layer whose parameters are differentiable and can therefore be part of a larger network, including the feature aggregation and prediction layers, which can be trained end-to-end via gradient back-propagation (Section 4). To address **C2**, we employ graph neural networks as the global inference model, where the graph corresponds to the explicit or implicit relational structure of the data owners. As such a graph might not be given in advance, we treat the combinatorial graph structure as a random variable of a product of Bernoulli distributions whose (differentiable) parameters can also be optimized via gradient-based approach (Section 5). The technical contributions of this work are summarized below.

1. We formalize a feature fusion perspective for distributed learning, in settings where data is vertically partitioned. This is an alternative view to VFL but as elaborated above, is more applicable when iterative training synchronicity is not possible among clients (Section 2).

2. We formulate a federated feature fusion (F^3) framework that consists of a network of pre-trained local models and a central model that collects and fuses the local feature representations (induced from these pre-trained models) to generate a global model with better predictive performance (Section 3). This is achieved via addressing **C1** (Section 4) and **C2** (Section 5) above.

3. We demonstrate experiments with four real-life data sets, including power grids and traffic networks, and show the effectiveness of the proposed framework (Section 7).

2 PROBLEM FORMULATION

Federated Feature Fusion (F^3) is a new but more practical setup for VFL [Hu et al., 2019, Chen et al., 2020]; it aims to enable collaboration between data owners that possess private access to different sets of features describing the same set of training data points. However, unlike VFL which require clients to synchronize their training processes [Yang et al., 2019b, Li et al., 2021, Fu et al., 2021, Cheng et al., 2021, Hu et al., 2019, Diao et al., 2021] in multiple iterations of communication, F^3 allows data owners to train their own local models in isolation and only requires one round of communication in which local feature representations induced from the heterogeneously pre-trained local models are shared with a trusted server for feature fusion.

Relation to FL with Heterogeneous Clients. We note that similar ideas on extending federated learning to accommodate clients with heterogeneous models [Tan et al., 2022b,a, Lin et al., 2020, Chen et al., 2022] has been proposed. However, these methods are still restricted to horizontal settings. Local models still need to operate on the same feature space and some of which also require client models to be trained together via multiple rounds of communication. As such, their focuses are on addressing different forms of heterogeneities: (1) heterogeneous data distributions; (2) heterogeneous model architectures; and (3) heterogeneous pre-training, which are all important but are different from feature heterogeneities, which is a new form of heterogeneities we are seeking to address.

To further emphasize on the novelty of our setting and solution significance, we review and discuss the formulation of VFL and F^3 below, which argues with concrete, real-life examples why the F^3 setting is more practical and how this practicality would entail significant technical challenges that necessitate new solutions in Sections 4 and 5.

Federated Learning with Vertically Partitioned Data. From a data perspective, the decentralized nature of data in VFL is a transposition to that of the traditional horizontal federated learning (HFL) [McMahan et al., 2017]. Instead of owning the same set of features for different sets of data points as in HFL, the data owners in VFL now own different sets of features for the same set of data points; and they share a common label set of these data points.

From the existing literature, two lines of work are noted. One takes the data matrix literally – by assuming tabular data and studying linear models – where model parameters have natural correspondence to the data parts [Hardy et al., 2017, Nock et al., 2018, Heinze et al., 2014, 2016]. Often, these approaches are hard to generalize to complex data with many owners. Another line of work advocates the use of models with modular structure in which separate parts of the model are responsible to locally aggregate different sets of local features owned by different owners; and a global parameterization is used to combine these local features. This is similar in spirit to F^3 but require clients to synchronize the training processes of their assigned model parts, which incurs expensive communication and creates dependence among the clients [Hu et al., 2019, Chen et al., 2020].[2]

Mathematically, for each datum $\mathbf{x}_k$ with label y_k, let $\mathbf{x}_k^i$ be the feature set of the datum that the i-th owner possesses. That is, $\mathbf{x}_k = (\mathbf{x}_k^1, \mathbf{x}_k^2, \ldots, \mathbf{x}_k^n)$ with n data owners. VFL aims to find aggregation parameter $\mathbf{w}$ and local representation parameters $\{\boldsymbol{\theta}_i\}_{i=1}^n$ that minimize

$$\mathbf{L}(\mathbf{w}, \boldsymbol{\theta}) \triangleq \frac{1}{m} \sum_{k=1}^{m} \ell\left[g\left(\left\{\phi_i\left(\mathbf{x}_k^i; \boldsymbol{\theta}_i\right)\right\}_{i=1}^n; \mathbf{w}\right), y_k\right] \quad (1)$$

where each $\phi_i(\mathbf{x}_k^i; \boldsymbol{\theta}_i)$ is a (learnable) local embedding of $\mathbf{x}_k^i$ parameterized by a separate parameter vector $\boldsymbol{\theta}_i$ owned by the i-th owner, $g(\phi_1, \phi_2, \ldots, \phi_n; \mathbf{w})$ is an aggregation function parameterized by $\mathbf{w}$ and ℓ is a prediction loss, e.g. the cross-entropy loss for classification or ℓ_2 loss for regression. The loss in Eq. (1) is averaged over all training data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$.

Federated Feature Fusion. The setting of F^3 is similar to VFL, except that the data owners share neither data nor models with each other to ensure a higher degree of privacy compliance, which is often the more practical setting in industry – see the example on power grid at the end of this section. For this reason, the VFL minimization task in Eq. (1) above is changed to finding a single set of aggregation parameter $\mathbf{w}$ that minimizes

$$\mathbf{L}(\mathbf{w}) \triangleq \frac{1}{m} \sum_{k=1}^{m} \ell\left[g\left(\mathbf{h}_k^1, \mathbf{h}_k^2, \ldots, \mathbf{h}_k^n; \mathbf{w}\right), y_k\right] \quad (2)$$

where $\mathbf{h}_k^i = \phi_i^*(\mathbf{x}_k^i)$ with $\phi_i^* = \arg\min_{\phi_i} \ell_i\left(\phi_i(\mathbf{x}_k^i), y_k\right)$ which characterizes the locally optimal feature representation obtained in isolation by the i-th owner. As such, Eq. (2) only requires one round of communication where $\{\mathbf{h}_k^i\}_{k,i}$ are communicated to a trusted server. Prior to that, each data owner can freely learn their own feature representation model $\phi_i(\mathbf{x}_k^i)$ with different parameterization and architecture, catering towards their own compute capacities and data representation. This avoids forcing the data owners to participate in a joint training scheme which often requires expensive coordination and is not practical.

However, in exchange for this practicality, two key challenges arise. First, as local models are separately trained, the correspondence between components of induced feature representations across local models become ambiguous since there is no mechanism to enforce their alignment. Second, for the same reason, there are potential innate local interactions among subsets of clients and a naive concatenation or averaging of their corresponding feature presentations will likely ignore such interactions, resulting in decreasing performance. These correspond to high-level challenge **C1** and **C2** in Section 1 which will be addressed in Sections 4 and 5 as our key technical contributions.

Separation of Local and Global Model Training. We remark that the separation of the local and global model training is driven by practical constraints in the real world. For example, there are cases in which data owners prefer to only release pre-trained models for the collaboration, which are not updatable. This is often preferable in production systems that compartmentalize into separately pre-trained

[2]Note that the approach proposed by Hu et al. [2019] assumes no parameters for the global model. Were global parameters present, gradient communication is inevitable.

workflows maintained by different product groups [Su et al., 2018], which needs a decoupled architecture such that updates to such workflow models can be implemented fast to scale the business. A similar design appears in many previous works, including Wang et al. [2022], Lam et al. [2021], Yurochkin et al. [2019b], Hoang et al. [2019a], Yurochkin et al. [2019a], Hoang et al. [2019b, 2020].

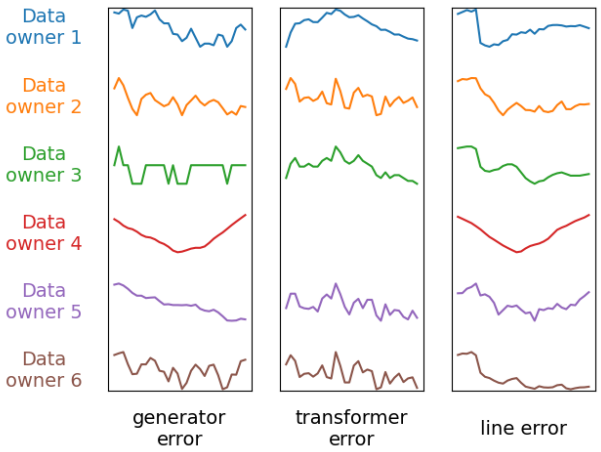

Figure 1: Federated Feature Fusion: A global prediction is produced collectively based on a set of global features which are the result of fusing local feature representations supplied by the data owners. These feature representations are induced from locally trained models on raw local data which might be heterogeneous.

Data Example. Let us consider the power grid monitoring task as an example. Figure 1 visualizes PMU measurements distributed across data owners. A panel of time series corresponds to a specific time window and the series collectively represent one data point, which the event detection system classifies. In this simplified illustration, each data owner possesses one series recorded by one PMU; but in practice they may own different amounts of PMUs (and thus series). Moreover, the series may differ in length because of varying sampling frequencies; and the series are multivariate with possibly different number of variates. All these variations contribute to feature heterogeneity, which necessitates the construction of separate local models. Note that if an event does not cascade over the entire grid, some local models may report event whereas others report normal, resulting in conflicting opinions. A consensus global model is responsible to resolve the conflict. Additionally, missing data may occur.

3 FEDERATED FEATURE FUSION

As detailed above, the proposed framework for F^3 consists of local models ϕ_i and a global feature fusion model g, such that their composition minimizes the loss in Eq. (2). Each data owner i possesses a local model trained with its data, independently of other owners. This way, no data sharing is invoked and privacy is of minimal concern. However, because the local models lack a global vision and may be conflicting, a central (global) model is key to coordinating the local opinions for final prediction. To maintain autonomy, local models are frozen once pre-trained and will not join the training of the global model. Data owners send local data representations to a centralized server for global model training (and inference). In other words, the global model queries neither the raw data nor the local models from data owners. As long as owners agree to send the less decipherable representations to the central server, global inference can be made.

Local Models. We treat a neural network except the final output layer as a feature extractor, which produces the representation $\mathbf{h}_k^i$ of an input fragment $\mathbf{x}_k^i$; and treat for simplicity the output layer as a logistic regression. That is, a local model $g_i(\mathbf{x}_k^i)$ reads:

$$g_i\left(\mathbf{x}_k^i\right) \triangleq \operatorname{softmax}\left(\mathbf{W}_i \cdot \mathbf{h}_k^i + \mathbf{b}_i\right) \tag{3}$$

where $\mathbf{h}_k^i = \phi_i\left(\mathbf{x}_k^i\right)$. Hereafter, we will interchangeably use *representation*, *embedding*, and *latent vector* to mean $\mathbf{h}_k^i$. These $\mathbf{h}_k^i$'s are assumed to have the same shape across i, although $\mathbf{x}_k^i$ can have different shapes and the embedding function can have different architectures to cope with feature heterogeneity. A simple example of the embedding function is a fully connected layer $\mathbf{h}_k^i = \mathrm{ReLU}(\mathbf{U}_i \cdot \mathbf{x}_k^i + \mathbf{c}_i)$; but an arbitrarily complex function is also applicable.

Global Model. The global model g melds together all local representations to generate a prediction:

$$y_k \simeq \widehat{y}_k \triangleq g\left(\mathbf{h}_k^1, \mathbf{h}_k^2, \ldots, \mathbf{h}_k^n; \mathbf{w}\right) \tag{4}$$

which is parameterized by $\mathbf{w}$. For example, the parameterization $\mathbf{w} = \{\mathbf{W}_0, \mathbf{W}_1, \mathbf{b}_0, \mathbf{b}_1\}$ characterize two fully connected (FC) layers interleaved with mean pooling:

$$g\left(\left\{\mathbf{h}_k^i\right\}_{i=1}^n; \mathbf{w}\right) = \operatorname{softmax}\left(\mathbf{W}_1 \cdot \boldsymbol{\alpha} + \mathbf{b}_1\right) \text{ where}$$
$$\boldsymbol{\alpha} = \frac{1}{n}\sum_{i=1}^n \mathrm{ReLU}\left(\mathbf{W}_0 \cdot \mathbf{h}_k^i + \mathbf{b}_0\right) \tag{5}$$

Given a particular parameterization $\mathbf{w}$, we can substantiate Eq. (4) above and plug it into Eq. (2). The optimal value for $\mathbf{w}$ can then be achieved by solving the corresponding minimization task therein. However, designing the form of $\mathbf{w}$ is highly non-trivial and is in fact tied to the previously mentioned challenges **C1** and **C2**, as elaborated below.

Challenges. Two considerations are pertinent to the design of $\mathbf{w}$. First, when the latent dimensions have semantic meaning – e.g. when the local models are trained to yield disentangled representations [Higgins et al., 2018] – each latent feature of the local representations may not match, because

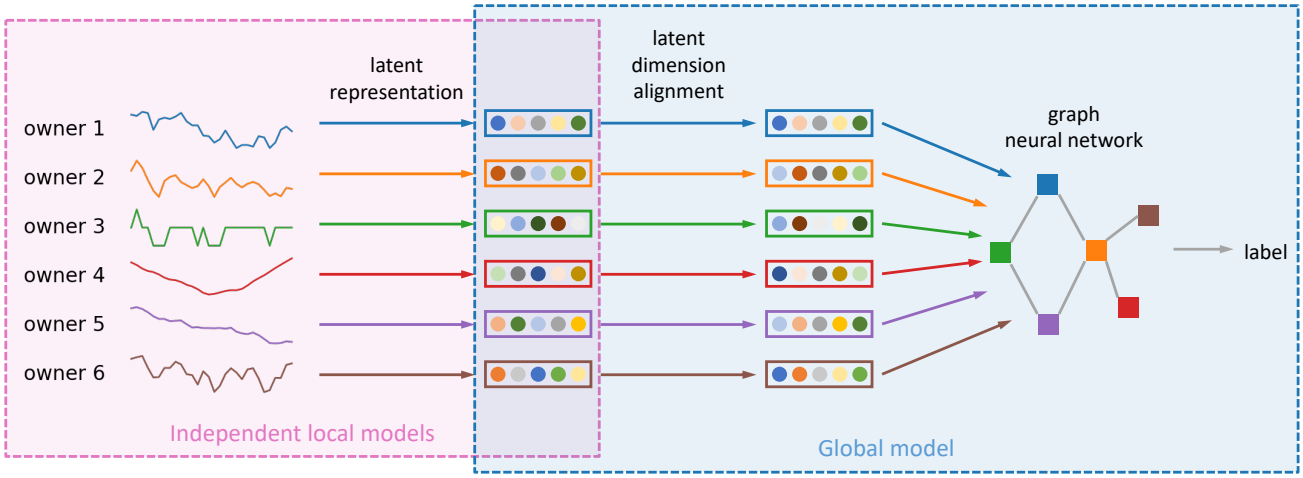

Figure 2: Federated Feature Fusion Framework. Local models are trained independently and separately from the global model. The algorithm is summarized in Algorithm 1.

an arbitrary permutation of the latent dimensions does not change a local model. Second, a naive mean pooling as in (5) misses the inter-dependencies between local data, leading to a less well performing global model. Such inter-dependencies occur in the power grid example because of the physics of an electricity network. Hence, we use latent alignment to address the first problem and graph neural network to address the second one. Incorporating these two components, we show the full proposed framework in Figure 2 and Algorithm 1. The solutions to these challenges will be discussed in Sections 4 and 5.

Algorithm 1 Federated Feature Fusion (F^3)

function TRAINING($\{(\mathbf{x}_k^i, y_k)_{k=1}^m\}_{i=1}^n$)
  Local clients learn $\{g^i\}_{i=1}^n$ with $\{(\mathbf{x}_k^i, y_k)_{k=1}^m\}_{i=1}^n$
  Local clients share $\{\mathbf{h}_k^i \triangleq \phi_i(\mathbf{x}_k^i)\}_{i=1}^n$ via Eq. (3)
  Minimize Eq. (2) composed with Eq. (8)
  Minimizing loss is averaged over samples of $\widehat{\mathbf{A}}$
  Entries of $\widehat{\mathbf{A}}$ are sampled using (9)
end function

function INFERENCE($\{\mathbf{x}_k\}_{k=1}^m$ where $\mathbf{x}_k = (\mathbf{x}_k^i)_{i=1}^n$)
  Local clients evaluate and send $\{\mathbf{h}_k^i\}_{i,k}$ to server
  Prediction is produced via Eq. (5)
end function

4 ALIGNING REPRESENTATIONS

For the global model to be meaningful, the feature dimensions of the local representations $\mathbf{h}_k^i$ should be aligned under the same feature space. For example, in (5), all $\mathbf{h}_k^i$'s multiply the same weight matrix $\mathbf{W}_0$: each element of $\mathbf{h}_k^i$ thus corresponds to one input neuron of the initial fully connected layer. Permutations of elements will destroy the correspondence: Arbitrary arrangements of feature dimensions of the latent vectors cause ambiguity on what an optimal global model can be built.

To elaborate, let us use a vector p to denote the index (column) permutation of a vector (matrix). Then, the ith local model (3) can be equivalently written as

$$g_i\left(\mathbf{x}_k^i\right) \triangleq \mathrm{softmax}\left(\mathbf{W}_i\left[:, \mathrm{p}_i\right] \cdot \mathbf{h}_k^i\left[\mathrm{p}_i\right] + \mathbf{b}_i\left[\mathrm{p}_i\right]\right) \quad (6)$$

where $\mathbf{h}_k^i \triangleq \phi_i\left(\mathbf{x}_k^i\right)$. This is true for any permutation p_i as long as the embedding function is able to produce a permuted $\mathbf{h}_k^i[\mathrm{p}_i]$ under the same input $\mathbf{x}_k^i$. Such a requirement can be easily satisfied if the embedding function is a fully connected layer such as $\mathbf{h}[\mathrm{p}] = \mathrm{ReLU}(\mathbf{W}[\mathrm{p}, :] \cdot \mathbf{x} + \mathbf{b}[\mathrm{p}])$. In fact, it is satisfied by most neural networks as well. The supplement gives another example: GRU [Cho et al., 2014].

Hence, we propose to align the feature dimensions across all local vectors $\mathbf{h}_k^i$ to disambiguate the ambiguity. This proposal amounts to adapting the global model (4) to:

$$y_k \simeq \widehat{y}_k \triangleq g\left(\mathbf{P}_1 \cdot \mathbf{h}_k^1, \mathbf{P}_2 \cdot \mathbf{h}_k^2, \ldots, \mathbf{P}_n \cdot \mathbf{h}_k^n\right), \quad (7)$$

where $\mathbf{P}_i$ is an alignment matrix for each data owner i, implementing the (manual) index or column permutation above in linear algebra. We can then treat each $\mathbf{P}_i$ as a free parameter matrix to optimize. It may be square or rectangle, the latter case indicating a change of the number of features. We also show an alternative hard alignment by parameterizing $\mathbf{P}_i$ as a permutation matrix in the supplement.

5 LEARNING A CONSENSUS GRAPH

The example global model (5) performs a naive averaging for the local representations. Since data owners are often interconnected, a more expressive model is needed to exploit their relational interactions to improve inference [Battaglia et al., 2018]. Here, we use a graph neural

network (GNN) [Zhang et al., 2020, Wu et al., 2021] to model and learn these high-order relational interactions.

A. Modeling Consensus via GCN with Latent Graph. All variants of GNN are applicable to our setting but we focus on the most basic GCN [Kipf and Welling, 2017] for presentation clarity. Let $\mathbf{A}$ be the graph's adjacency matrix and let $\mathbf{H}_k$ be the matrix of aligned local representations:

$$\mathbf{H}_k \triangleq \begin{bmatrix} -(\mathbf{P}_1\mathbf{h}_k^1)^\top- \\ \vdots \\ -(\mathbf{P}_n\mathbf{h}_k^n)^\top- \end{bmatrix}.$$

The global prediction $y_k \simeq \widehat{y}_k = g(\mathbf{h}_k^1, \ldots, \mathbf{h}_k^n; \mathbf{w})$ in Eq. (4) is then substantiated with

$$g\left(\left\{\mathbf{h}_k^i\right\}_{i=1}^n; \mathbf{w}\right) \triangleq \operatorname{softmax}\left(\frac{1}{n}\mathbf{1}^\top \mathbf{A}^\dagger \cdot \mathbf{W}_1\right) \quad \text{with}$$
$$\mathbf{A}^\dagger = \widehat{\mathbf{A}} \cdot \operatorname{ReLU}\left(\widehat{\mathbf{A}}\mathbf{H}_k\mathbf{W}_0\right) \tag{8}$$

where $\widehat{\mathbf{A}}$ is a normalization of $\mathbf{A}$ [Kipf and Welling, 2017] and $\mathbf{w} = (\mathbf{W}_0, \mathbf{W}_1, \{\mathbf{P}_i\}_{i=1}^n)$ are learnable parameters.

Here, we adapt the traditional GCN prediction with the inclusion of $\frac{1}{n}\mathbf{1}^T$ as pooling before output. Modulo this modification, Eq. (8) is the traditional one used in the literature with bias terms omitted. It is interesting to note the equivalence between the GCN (8) and graph-agnostic (5) models when $\widehat{\mathbf{A}}$ is replaced by $\mathbf{I}$, omitting the bias terms.

In GCN, $\mathbf{A}$ corresponds to the consensus graph among local owners as graph nodes. If such a graph is not present, it is possible to learn one such that (8) still outperforms (5). In this case, we treat $\mathbf{A}$ as a random variable of the matrix Bernoulli distribution, where the success probabilities are free parameters to learn. Formally, the elements $\mathbf{A}_{ij}$ are independent and each follows $\operatorname{Ber}(\theta_{ij})$, where θ_{ij} denotes the corresponding probability [Kipf et al., 2018, Shang et al., 2021]. Then, the global model g has $\mathbf{W}_0$, $\mathbf{W}_1$, the $\mathbf{P}_i$'s, as well as θ, as parameters. Following Franceschi et al. [2019], Shang et al. [2021], we formulate the training loss as an expectation over $\mathbf{A}$'s distribution and draws a sample $\mathbf{A}$ to obtain an unbiased estimate of the loss and its gradient.

Security in Transmitting Data Representation. Transmitting data representation might risk exposing raw data and mitigating such risks in multiple rounds of communication is often non-trivial. However, in our case, there is a single communication round so sanitizing the data representation does not pose a new challenge. It can be addressed using a variety of existing, well-established techniques such as Shamir's Secret Sharing (SSS) [Shamir, 1979]. Each local representation (a secret) can be splitted into multiple shares distributed to multiple central entities who process and combine the results to reproduce the desired result. As the central entities process the shares independently and only communicate the results to a coordinator, the SSS protocol can guarantee no single central entity would have enough information to access any local representations.

B. Differentiable Graph Sampling. However, the central challenge of this approach is that the sample $\mathbf{A}_{ij}$ is not differentiable with respect to the corresponding Bernoulli bias θ_{ij}, which in turn makes the training loss non-differentiable with respect to θ. To sidestep this difficulty, we propose the following reparameterization, which presents a learnable (differentiable) transformation of a sample drawn from a continuous distribution to a discrete Bernoulli sample. This transformation is detailed in Definition 1 below, which is followed by Theorem 1 showing the distributional convergence of this transformation to the desired distribution.

Definition 1. Let F be the CDF of an arbitrary continuous probability distribution. Sample s from this **reference distribution** and let

$$z \triangleq \operatorname{sigmoid}\left(\frac{1}{\tau}\left(F^{-1}(\theta) - s\right)\right), \quad \tau > 0. \tag{9}$$

We call this the **ICDF** reparameterization which is named after the use of inverse cumulative F^{-1}.

Theorem 1. For all $\tau > 0$, $\theta \in (0, 1)$ and $t \in [0, 1]$, if the distribution with CDF F is finitely supported on $[a, b]$, then

$$\Pr(z \le t) = \begin{cases} 0 & \text{if } t < \sigma((F^{-1}(\theta) - b)/\tau), \\ 1 & \text{if } t > \sigma((F^{-1}(\theta) - a)/\tau) \end{cases} \tag{10}$$

or otherwise,

$$\Pr(z \le t) = 1 - F\left(F^{-1}(\theta) + \tau \log\left(\frac{1}{t} - 1\right)\right). \tag{11}$$

In case the distribution is not finitely supported (i.e., $a = -\infty$ and/or $b = +\infty$), Eqs. (10) and (11) still hold because either (or both) of the first two cases will not be invoked. Thus, the distribution of z converges to $\operatorname{Ber}(\theta)$ as $\tau \to 0$.

Discussion. We note that an alternative to the above can be achieved via using the Gumbel softmax reparameterization [Jang et al., 2017, Maddison et al., 2017] which also features a differentiable relaxation of the Bernoulli distribution that approximates it asymptotically. However, in order to obtain one Bernoulli sample, the Gumbel trick requires to sample the Gumbel distribution twice.

Instead, our proposed reparameterization only requires sampling from the reference distribution only once. We also show that the **ICDF** reparameterization converges as fast as the Gumbel softmax. Both approaches have asymptotic convergence rate on the order of $O(\tau^2)$ as shown in the supplement. Empirically, we also show that **ICDF** induces marginally better performance than Gumbel softmax. This is why we prefer **ICDF** to Gumbel in our work.

6 RELATED WORK

The concept of federated learning was first coined by McMahan et al. [2017] and it has attracted surging interests since then. Recent literature reviews [Yang et al., 2019a, Li et al., 2019, Kairouz et al., 2019] have comprehensively studied the topic, summarized systems and infrastructures, and also suggested open problems. Among these, one interesting direction is a new family [Hardy et al., 2017, Nock et al., 2018, Heinze et al., 2014, 2016] of federated learning that studies a federated scenario where features (instead of samples) are split across owners. This setting bears resemblance to our federated feature fusion scenario, but a key distinction is all literature in this direction focuses on local models that can be trained together whereas in our scenarios, local models are trained in isolation to avoid the cost and overhead of coordination among different parties.

To build consensus among local models, our framework learns parameter matrices to align their local representations. Such alignments similarly appear in model fusion, where a number of models are fused together into a common model through aligning model parameters [Yurochkin et al., 2019a]. In the context of deep learning, if the neural networks come from the same model family, their weights can be matched layer-wise, even if the numbers of weights are different [Yurochkin et al., 2019b, Wang et al., 2020]. The referenced work treats the problem as a bipartite graph matching, where the cost matrix is inferred from maximum a posteriori estimation. Then, the Hungarian algorithm [Kuhn, 1955] is applied to find the matching. In our work, instead we treat the permutation alignment as a differentiable parameterization with the help of Sinkhorn–Knopp [Sinkhorn and Knopp, 1967, Mena et al., 2018, Emami and Ranka, 2018], so that it can be learned end-to-end with other parameters.

Our framework also advocates learning a graph of data owners in the global model. Graph structure learning appears under various contexts. One field of study is grounded in the context of probabilistic graphical models, whereby a directed acyclic structure is learned. Gradient-based approaches in this context include Zheng et al. [2018], Yu et al. [2019], Lachapelle et al. [2020]. On the other hand, a general graph may still be useful without resorting to causality. Recent approaches supporting GNN-based modeling include Kipf et al. [2018], Franceschi et al. [2019], Wu et al. [2020], Shang et al. [2021], wherein a graph structure is simultaneously learned together with the GNN parameters.

7 EXPERIMENTS

This section reports comprehensive experiment results to demonstrate the effectiveness of our proposed federated feature fusion (F^3) substantiated with the developed techniques in Sections 4 and 5.

Table 1: Dataset Statistics.

	METR-LA	PEMS-BAY	PMU-B	PMU-C
# samples	2856	4343	4853	1884
# owners	207	325	43	188
# features	1	1	2	2
# classes	2	2	4	4
series length	12	12	30	30
missing data	N	N	Y	Y
given graph	Y	Y	N	N

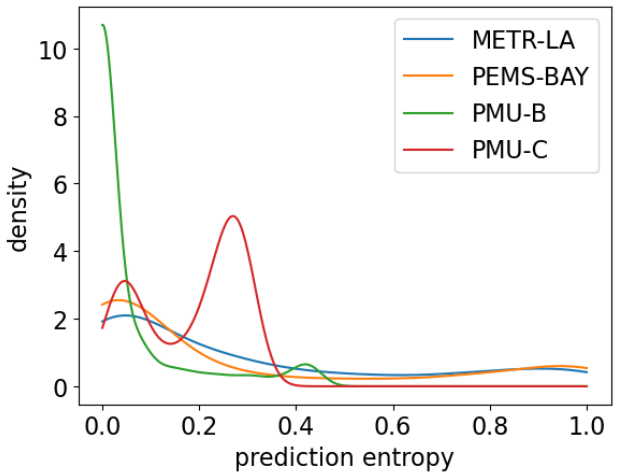

Figure 3: Distributions of prediction entropy.

Datasets. We use four real-life, time series datasets. Two are PMU data collected from multiple data owners of the U.S. power grid. For proof of concept, we smooth out heterogeneity and prepare homogeneous data sets. Such a preprocessing is sufficient to test the proposed techniques under minimal impact of the complication by the otherwise diverse local models. Since the PMU data sets are proprietary, we also use two public, traffic data sets [Li et al., 2018] for experimentation. A summary of these data sets is given in Table 1 while other processing details are deferred to the supplement due to limited space.

Experiment Setting. All local models are LSTM [Hochreiter and Schmidhuber, 1997] with the same hyperparameters, but pre-trained separately by using local data. The local models are not fine-tuned in the training of the global model. Each dataset is split randomly for training/validation/testing. See the supplement for further details.

Conflicting Local Predictions. We first show that local models do not produce consistent predictions, which rationalizes the effort of training a global model and performing federated feature fusion. For each datum, we compute the entropy of the predicted labels and summarize the entropies for all data into a distribution, plotted in Figure 3. Recall that the lower the entropy, the more consistent the local

Table 2: Effectiveness of latent alignment in a graph-based global model. Superscript numbers are standard deviations.

	METR-LA		PEMS-BAY		PMU-B		PMU-C	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC
A: Horizontal FL	$.25^{.000}$	-	$.33^{.000}$	-	$.36^{.000}$	-	$.29^{.000}$	-
B: Majority Voting	$.11^{.000}$	-	$.09^{.000}$	-	$.29^{.000}$	-	$.18^{.000}$	-
C: Binary Thresholding	$.69^{.000}$	-	$.64^{.000}$	-	-	-	-	-
D: Best Model Selection	$.53^{.000}$	$.70^{.000}$	$.55^{.000}$	$.79^{.000}$	$.37^{.000}$	$.69^{.000}$	$.32^{.000}$	$.62^{.000}$
E: Mean Pooling – Eq. (5)	$.77^{.009}$	$.96^{.004}$	$.74^{.012}$	$.93^{.001}$	$.38^{.008}$	$.71^{.006}$	$.34^{.008}$	$.64^{.010}$
F: Transformer	$.78^{.023}$	$.94^{.018}$	$.72^{.045}$	$.93^{.027}$	$\mathbf{.39}^{.003}$	$.70^{.009}$	$.40^{.053}$	$.67^{.058}$
G: Concatenation	$\mathbf{.83}^{.008}$	$\mathbf{.97}^{.002}$	$.80^{.066}$	$.96^{.028}$	$\mathbf{.39}^{.006}$	$.71^{.036}$	$.40^{.025}$	$.68^{.040}$
H: F^3 w. no alignment	$.80^{.009}$	$.96^{.004}$	$.75^{.009}$	$.94^{.001}$	$\mathbf{.39}^{.003}$	$.73^{.015}$	$.40^{.020}$	$.66^{.018}$
J: F^3 w. parameter tying	$.82^{.009}$	$\mathbf{.97}^{.001}$	$.75^{.009}$	$.94^{.004}$	$\mathbf{.39}^{.006}$	$.72^{.010}$	$.37^{.012}$	$.66^{.008}$
K: F^3 w. alignment	$\mathbf{.83}^{.010}$	$\mathbf{.97}^{.001}$	$\mathbf{.86}^{.005}$	$\mathbf{.98}^{.002}$	$\mathbf{.39}^{.008}$	$.73^{.008}$	$\mathbf{.45}^{.015}$	$.72^{.003}$
L: VFL w. graph/alignment	$\mathbf{.83}^{.012}$	$\mathbf{.97}^{.001}$	$\mathbf{.86}^{.014}$	$\mathbf{.98}^{.002}$	$\mathbf{.39}^{.006}$	$\mathbf{.74}^{.009}$	$\mathbf{.45}^{.015}$	$\mathbf{.73}^{.003}$
M: VFL w.o. pre-trained local	$.77^{.020}$	$.94^{.021}$	$.77^{.014}$	$.95^{.006}$	$.34^{.014}$	$.69^{.012}$	$.35^{.008}$	$.65^{.014}$

Table 3: Impact of learning a graph across different alignment settings.

		METR-LA		PEMS-BAY		PMU-B		PMU-C	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC
No Align	No Graph	$.768^{.009}$	$.957^{.004}$	$.738^{.012}$	$.935^{.001}$	$.381^{.008}$	$.711^{.006}$	$.342^{.008}$	$.636^{.010}$
	Given Graph	$.763^{.020}$	$.957^{.007}$	$.742^{.024}$	$.942^{.005}$	-	-	-	-
	κ-NN Graph	$.715^{.015}$	$.952^{.004}$	$.695^{.013}$	$.934^{.004}$	$.372^{.001}$	$.711^{.013}$	$\mathbf{.404}^{.016}$	$\mathbf{.680}^{.014}$
	ICDF	$\mathbf{.798}^{.009}$	$\mathbf{.963}^{.004}$	$\mathbf{.755}^{.009}$	$\mathbf{.943}^{.001}$	$\mathbf{.387}^{.003}$	$\mathbf{.734}^{.015}$	$.403^{.020}$	$.663^{.018}$
Align	No Graph	$.813^{.009}$	$.970^{.002}$	$.846^{.008}$	$.977^{.001}$	$.386^{.009}$	$.725^{.012}$	$.386^{.008}$	$.694^{.005}$
	Given Graph	$.828^{.007}$	$.974^{.001}$	$.854^{.003}$	$.977^{.001}$	-	-	-	-
	κ-NN Graph	$.803^{.020}$	$.968^{.002}$	$.855^{.003}$	$.973^{.002}$	$.378^{.002}$	$.718^{.015}$	$.418^{.007}$	$.702^{.009}$
	ICDF	$\mathbf{.835}^{.010}$	$\mathbf{.975}^{.001}$	$\mathbf{.860}^{.005}$	$\mathbf{.980}^{.002}$	$\mathbf{.390}^{.008}$	$\mathbf{.734}^{.008}$	$\mathbf{.451}^{.015}$	$\mathbf{.725}^{.003}$

predictions. The figure, however, shows that a substantial amount of entropies is away from zero, suggesting that local predictions are inconsistent.

Effectiveness of Federated Feature Fusion. We make two sets of comprehensive comparisons to evaluate the effectiveness of the proposed framework. The first set, as outlined in Table 2, compares F^3 with a number of non-graph baselines (**A–G**), including: (a) horizontal FL (**A**) which requires both model homogeneity and training synchronicity among clients that are not admitted in our setting; (b) a set of standard ensemble strategies (**B–F**) that combine the local models, such as voting, binary thresholding, best local model, mean-pooling via Eq. (5), as well as a simplified Set Transformer with 2 layers and 4 heads [Lee et al., 2019]; and (c) a vertical FL baseline (**G**) via feature concatenation.

This set also contains several variants of our proposed federated feature fusion model, featuring an ablation study of the effectiveness of our model components: (**H**) F^3 without alignment; (**J**) F^3 with partially tied parameters among local models; and (**K**) F^3 with learnable alignment. Note that **J** is an alternative to alignment which comes with the cost of imposing strong homogenization – though not as strong as **A** – among local models despite the different nature of their data. All variants **H–K** use **ICDF** reparameterization to learn the graph structure.

From Table 2, we observe that baselines **A–D**, lacking either local models or a holistic global model, perform significantly worse than the other baselines (including our F^3 variants, the ensemble via mean pooling, and concatenation). On the other hand, baselines **E–G** perform better than **A–D** but they lack a proper alignment of local models or they impose a strong form of homogenization among local models to sidestep alignment. Therefore, they are expectedly outperformed by **K** that performs alignment.

We also compare with two variants of our model **K** in the vertical FL setting. **L** uses the same local and global models as **K** but allows gradients to be sent back to local clients, thus local models can be updated. It achieves similar performance as **K** but leads to much more communication cost with multiple rounds of gradient messages. **M** assumes no local pre-trained models and all local and global models are trained jointly from scratch. Its performance is much worse

than **K** and **L**, which explains the merit of pre-trained local models. Another typical VFL baseline **G** with pre-trained local models and a simple concatenation based global model is also inferior.

Impact of Learning a Graph. Our next set of experiments, as outlined in Table 3, demonstrate the impact of learning a graph that characterizes the innate local interactions among subsets of clients, following our challenge statement **C2** in the introduction, on both alignment and non-alignment baselines. This provides ablation studies on the isolated impact of having a specific graph learning component. In particular, for each alignment setting, we demonstrate the impact on performance with (a) not using a graph; (b) using a predefined graph; (c) learning the graph structure using a κ-**NN** baseline [Fatemi et al., 2021]; and (d) learning the graph structure using **ICDF**. The reported numbers suggest that regardless of whether the model performs alignment, graph learning always improves performance.

Remark. The κ-**NN** baseline ($\kappa = 10$) is implemented following the description in [Fatemi et al., 2021]. Specifically, during training, we generate a local graph for each batch for node features $\mathbf{X}$ via a symmetrization of $\tilde{\mathbf{A}} = \kappa\text{-}\mathbf{NN}(\mathrm{MLP}(\mathbf{X}))$ which (1) feeds the node features through a MLP neural block; and (2) draws an edge between each node and its κ nearest neighbors where the neighborhood is defined using the cosine similarity on the space of MLP-projected feature vectors.

8 CONCLUSION

In this paper, we study federated feature fusion, which presents a less addressed scenario of federated learning where data owners or clients need to customize their own local models to accommodate different sets of features. Unlike federated learning, the clients need to learn their own model separately in isolation and only communicate their local feature representations afterwards. We motivate the practicality of federated feature fusion with a power grid example and propose a local–global model framework for it. Two important components of the framework are the alignment of the data representations produced by local models and the learning of the global model by using a graph neural network. Comprehensive experiments suggest the feasibility and the effectiveness of federated feature fusion. We release our code at `https://github.com/matenure/federated_feature_fusion`.

Acknowledgements

This material is based upon work supported by the Department of Energy under Award Number(s) DE-OE0000910, and the Defense Advanced Research Projects Agency (DARPA) through Cooperative Agreement D20AC00004 awarded by the U.S. Department of the Interior (DOI), Interior Business Center. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

Jan Philipp Albrecht. How the GDPR will change the world. *European Data Protection Law Review*, 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. Preprint arXiv:1806.01261, 2018.

Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang. Personalized federated learning with graph. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. VAFL: a method of vertical asynchronous federated learning. In *ICML Workshop*, 2020.

Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *The International Conference on Learning Representations (ICLR)*, 2021.

Patrick Emami and Sanjay Ranka. Learning permutations with Sinkhorn policy gradient. Preprint arXiv:1805.07010, 2018.

Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, 2019.

Fangcheng Fu, Yingxia Shao, Lele Yu, Jiawei Jiang, Huanran Xue, Yangyu Tao, and Bin Cui. Vf2boost: Very fast vertical federated gradient boosting for cross-enterprise learning. In *Proceedings of the 2021 International Conference on Management of Data*, pages 563–576, 2021.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. Preprint arXiv:1711.10677, 2017.

Christina Heinze, Brian McWilliams, Nicolai Meinshausen, and Gabriel Krummenacher. LOCO: Distributing ridge regression with random projections. Preprint arXiv:1406.3469, 2014.

Christina Heinze, Brian McWilliams, and Nicolai Meinshausen. DUAL-LOCO: Distributing statistical estimation using random projections. In *AISTATS*, 2016.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. Preprint arXiv:1812.02230, 2018.

Quang Minh Hoang, Trong Nghia Hoang, Kian Hsiang Low, and Carl Kingsford. Collective model fusion of multiple black-box experts. In *Proc. ICML*, 2019a.

Trong Nghia Hoang, Quang Minh Hoang, Kian Hsiang Low, and Jonathan How. Collective online learning of Gaussian processes in massive multi-agent systems. In *Proc. AAAI*, 2019b.

Trong Nghia Hoang, Chi Thanh Lam, Kian Hsiang Low, and Patrick Jaillet. Learning task-agnostic embedding of multiple black-box experts for multi-task model fusion. In *Proc. ICML*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. FDML: A collaborative machine learning framework for distributed features. In *KDD*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G.L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. Preprint arXiv:1912.04977, 2019.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2): 83–97, 1955.

Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural DAG learning. In *ICLR*, 2020.

Chi Thanh Lam, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Patrick Jaillet. Model fusion for personalized learning. In *ICML*, 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*, 2021.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. Preprint arXiv:1907.09693, 2019.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with Gumbel-Sinkhorn networks. Preprint arXiv:1802.08665, 2018.

Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Entity resolution and federated learning get a federated resolution. Preprint arXiv:1803.04035, 2018.

A. Shamir. How to share a secret? *Communications of the ACM*, 22:612–613, 1979.

Chao Shang, Jie Chen, and Jianbo Bi. Discrete graph structure learning for forecasting multiple time series. In *ICLR*, 2021.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.*, 21(2):343–348, 1967.

Smartgrid.gov. Recovery act: Synchrophasor applications in transmission systems. Webpage retrieved in Jan 2021. https://www.smartgrid.gov/recovery_act/program_impacts/applications_synchrophasor_technology.html.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas. A re-ranker scheme for integrating large-scale NLU models. In *IEEE SLT*, 2018.

Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI Conference on Artificial Intelligence*, volume 1, page 3, 2022a.

Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *arXiv preprint arXiv:2209.10083*, 2022b.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *ICLR*, 2020.

Zhenyi Wang, Xiaoyang Wang, Li Shen, Qiuling Suo, Kaiqiang Song, Dong Yu, Yan Shen, and Mingchen Gao. Meta-learning without data via Wasserstein distributionally-robust model fusion. In *UAI*, 2022.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *KDD*, 2020.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2), 2019a.

Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019b.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In *ICML*, 2019.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, and Trong Nghia Hoang. Statistical model aggregation via parameter matching. In *NeurIPS*, 2019a.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*, 2019b.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *Transactions on Knowledge and Data Engineering*, 2020.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *NeurIPS*, 2018.