Variational Federated Multi-Task Learning

Luca Corinzia
Institute for machine learning
ETH Zurich
luca.corinzia@inf.ethz.ch

Joachim M. Buhmann
Institute for machine learning
ETH Zurich
jbuhmann@inf.ethz.ch

Abstract

In classical federated learning a central server coordinates the training of a single model on a massively distributed network of devices. This setting can be naturally extended to a multi-task learning framework, to handle real-world federated datasets that typically show strong non-IID data distributions among devices. Even though federated multi-task learning has been shown to be an effective paradigm for real world datasets, it has been applied only to convex models. In this work we introduce VIRTUAL, an algorithm for federated multi-task learning with non-convex models. In VIRTUAL the federated network of the server and the clients is treated as a star-shaped Bayesian network, and learning is performed on the network using approximated variational inference. We show that this method is effective on real-world federated datasets, outperforming the current state-of-the-art for federated learning.

1 Introduction

Large scale networks of remote devices like mobile-phones, wearables, smart-homes, self-driving cars and other IoT devices are becoming a significant source of data to train statistical models. This has generated growing interest to develop machine learning paradigms that can take into account distributed data-structure, despite of the several challenges arising in this setting:

Security Data generated by remote devices is often privacy-sensitive and its centralized collection and storage is governed by data protection regulations (e.g GDPR [43] and the Consumer Privacy Bill of Rights [19]). Learning paradigms that do not access user-data directly are hence desired.

System Remote devices in these networks have typically important storage and computational capacity constraints, limiting the complexity and the size of the model that can be used. Moreover, the communication of information between devices or between devices and a central server mostly happens on wireless networks and hence communication cost can become a significant bottleneck of the learning process.

Statistical The devices of the network typically generate samples with different user-dependent probability distributions, making the setting in general strongly non-IID. While it is a challenge to achieve high statistical accuracy for classical federated and distributed algorithms in this setting, a multi-task learning (MTL) approach can tackle heterogeneous data in a more natural way. Every device of the network requires a task-specific model, tailored for its own data distribution, to boost the performance of each individual task.

Federated learning (FL) [30] has emerged as the learning paradigm to address the scenario of learning models on private distributed data sources. It assumes a federation of devices called *clients* that both collect the data and carry out an optimization routine, and a *server* that coordinates the learning

Preprint. Under review.

by receiving and sending updates from and to the clients. This paradigm has been applied successfully in many real world cases, e.g to train smart keyboards in commercial mobile devices [46] and to train privacy-preserving recommendation systems [2]. Federated Averaging (`FedAvg`) [30, 33] is the state-of-the-art for federated learning with *non-convex* models and requires all the clients to share the same model. Hence it does not address the statistical challenge of strongly skewed data distributions, and while it has been shown to work well in practice for a range of (non-federated) real world datasets, it performs poorly in heterogeneous scenarios [30]. We address this problem introducing `VIRTUAL` (**V**ar**I**ational fede**R**aTed m**U**lti t**A**sk **L**earning), a new framework for federated MTL. In `VIRTUAL`, the central server and the clients form a Bayesian network and the inference is performed using variational methods. Every client has a task specific model that benefits from the server model in a transfer learning fashion with *lateral connections*. Hence a part of the parameters are shared between all clients, and another part is private and tuned separately. The server maintains a posterior distribution that represent the plausibility of the shared parameters. In one step of the algorithm, the posterior is communicated to the clients before the training starts, while during training the clients update the posterior given the likelihood of their local data. Finally, the posterior update is sent back to the central server.

Contributions Our main contributions are threefold: (i) We address for the first time the problem of federated MTL for *generic non convex models*, and we propose `VIRTUAL`, an algorithm to perform federated training with strongly non-IID client data distributions. (ii) We perform extensive experimental evaluation of `VIRTUAL` on real world federated datasets, showing that it outperforms the current state-of-the-art in FL and (iii) we frame the federate MTL problem as an inference problem in a Bayesian network bridging the frameworks of federated and transfer/continuous learning, which opens the door to a new class of application-specific federated algorithms.

2 The `VIRTUAL` algorithm

In FL, K clients are associated with K datasets $\mathcal{D}_1, \dots, \mathcal{D}_K$, where $\mathcal{D}_i = \{\mathbf{x}_i^{(n)}, y_i^{(n)}\}_{n=1}^{N_i}$ is in general generated by a client dependent probability distribution function (pdf) and only accessible by the respective client. It is natural to fit K different models, one for each dataset, enforcing a relationship between models using parameter sharing [11]. This approach has been investigated extensively, and it has been shown to boost effective sample size and performance in MLT for neural networks [35].

2.1 The Bayesian network

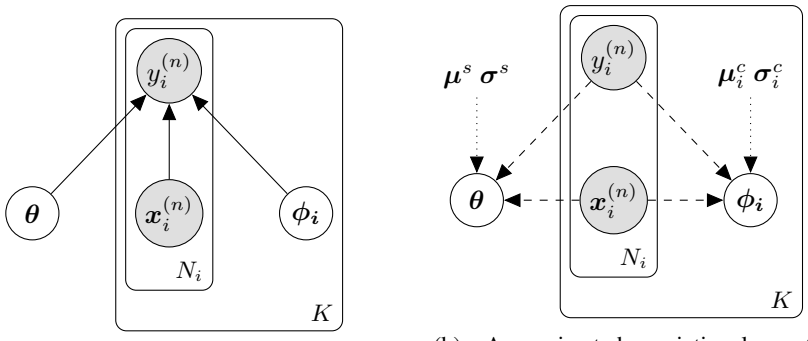
Assume a star-shaped Bayesian network with a server S with model parameters $\boldsymbol{\theta}$, as well as K clients with model parameters $\{\boldsymbol{\phi}_i\}_{i=1}^K$. Assume that every client is a discriminative model distribution over the input given by $p(y_i^{(n)}|\boldsymbol{x}_i^{(n)}, \boldsymbol{\theta}, \boldsymbol{\phi}_i)$ (a naive extension of the work could consider also generative models). Each dataset $\mathcal{D}_i$ is private to client i, hence it is not accessible to any other client or the central server S, and has a likelihood that factorizes as $p(\mathcal{D}_i|\boldsymbol{\theta}, \boldsymbol{\phi_i}) = \prod_{n=1}^{N_i} p(y_i^{(n)}|\boldsymbol{x}_i^{(n)}, \boldsymbol{\theta}, \boldsymbol{\phi}_i)$. Following a Bayesian approach, we assume a prior distribution over all the network parameters $p(\boldsymbol{\theta}, \boldsymbol{\phi_1}, \dots, \boldsymbol{\phi_K})$. The posterior distribution over all parameters, given *all* datasets $\mathcal{D}_{1:K} = \{\mathcal{D}_1, \dots, \mathcal{D}_k\}$ reads then

$$p(\boldsymbol{\theta}, \boldsymbol{\phi_1}, \dots, \boldsymbol{\phi_K}|\mathcal{D}_{1:K}) \propto \frac{1}{p(\boldsymbol{\theta})^{K-1}} \prod_{i=1}^{K} p(\boldsymbol{\theta}, \boldsymbol{\phi_i}|\mathcal{D}_i) \tag{1}$$

where we enforce that client-data is conditionally independent given server and client parameters, $p(\mathcal{D}_{1:K}|\boldsymbol{\theta}, \boldsymbol{\phi_1}, \dots, \boldsymbol{\phi_K}) = \prod_{i=1}^K p(\mathcal{D}_i|\boldsymbol{\theta}, \boldsymbol{\phi_i})$, and a factorization of the prior as $p(\boldsymbol{\theta}, \boldsymbol{\phi_1}, \dots, \boldsymbol{\phi_K}) = p(\boldsymbol{\theta}) \prod_{i=1}^K p(\boldsymbol{\phi_i})$. The Bayesian network is illustrated in Figure 1a.

2.2 The optimization procedure

The posterior given in Equation (1) is in general intractable and hence we have to rely on an approximation inference scheme (e.g. variational inference, sampling, expectation propagation [5]). Here we propose an expectation propagation (`EP`) like approximation algorithm [31] that has been shown



(a) Baysian network

(b) Approximated variational posterior model

Figure 1: Graphical models that describe the VIRTUAL framework for federated learning. The plates represent replicates. In both figures the outer plate replicates client i over the total number of clients K, while the inner plate replicates sample index n over the total number of samples per client N_i. Shadowed nodes represent observed variables and non-shadowed nodes represent latent variables. (a) Solid lines denote the discriminative model $p(y_i^{(n)}|\boldsymbol{x}_i^{(n)}, \boldsymbol{\theta}, \boldsymbol{\phi}_i)$. (b) Graphical model of the approximated variational posterior. Dashed lines denote (deterministic) dependencies in the approximated variational posterior while dotted lines denote stochastic dependencies. Here we indicate as $(\boldsymbol{\mu}^s,\boldsymbol{\sigma}^s)$ and $(\boldsymbol{\mu}_i^c,\boldsymbol{\sigma}_i^c)$ the collection of all Gaussian parameters of server and client i.

to be effective and to outperform other methods when applied in the continual learning (CL) setting [8, 32]. Let us denote the collection of all the client parameters by $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_K)$. Then we define a *proxy* posterior distribution that factorizes into a server and a client contribution for every client i as

$$q(\boldsymbol{\theta}, \boldsymbol{\phi}) = s(\boldsymbol{\theta})c(\boldsymbol{\phi}) = \left(\prod_{i=1}^{K} s_i(\boldsymbol{\theta})\right)\left(\prod_{i=1}^{K} c_i(\boldsymbol{\phi}_i)\right). \tag{2}$$

The fully factorization of both server and client parameters allows us in the following to perform a client update that is independent from other clients, and to perform a server update in the form of an aggregated posterior that preserves the privacy.

Given a factorization of this kind, the general EP algorithm refines one factor at each step. It first computes a refined posterior distribution where the refining factor of the proxy is replaced with the respective factor in the true posterior distribution. It then performs the update minimizing the Kullback-Leibler (KL) divergence between the full proxy posterior distribution and the refined posterior. The optimization to be performed for our particular Bayesian network and factorization is given by the following.

Proposition 1. *Assuming that at step t the factor i is refined, then the proxi pdf $s_i^{(t)}(\boldsymbol{\theta})$ and $c_i^{(t)}(\boldsymbol{\phi}_i)$ are found minimizing the variational free energy function*

$$\mathcal{L}(s_i(\boldsymbol{\theta}), c_i(\boldsymbol{\phi}_i)) = D_{KL}\left(s_i(\boldsymbol{\theta})\frac{s^{(t-1)}(\boldsymbol{\theta})}{s_i^{(t-1)}(\boldsymbol{\theta})}\middle|\middle|p(\boldsymbol{\theta})\frac{s^{(t-1)}(\boldsymbol{\theta})}{s_i^{(t-1)}(\boldsymbol{\theta})}\right) + D_{KL}\left(c_i(\boldsymbol{\phi}_i)||p(\boldsymbol{\phi}_i)\right) + \\ - \mathbb{E}_{\substack{s^{(t)}(\boldsymbol{\theta}) \\ c_i(\boldsymbol{\phi}_i)}} \log p(\mathcal{D}_i|\boldsymbol{\theta}, \boldsymbol{\phi}_i) \tag{3}$$

where $s^{(t)}(\boldsymbol{\theta}) = s_i(\boldsymbol{\theta}) \prod_{j\neq i}^{K} s_j^{(t-1)}(\boldsymbol{\theta})$ is the new posterior over server parameters.

Proof. At step t the global posterior for server parameters is $s^{(t)}(\boldsymbol{\theta}) = s_i(\boldsymbol{\theta}) \prod_{j\neq i}^{K} s_j^{(t-1)}(\boldsymbol{\theta})$ and analogously the client parameters distribution reads $c^{(t)}(\boldsymbol{\phi}) = c_i(\boldsymbol{\phi}_i) \prod_{j\neq i}^{K} c_j^{(t-1)}(\boldsymbol{\phi}_j)$. Then the EP-like update for the model described is given by minimizing the following KL divergence w.r.t

$s_i(\boldsymbol{\theta})$ and $c_i(\boldsymbol{\phi}_i)$

$$
\begin{aligned}
D_{KL}\left(s^{(t)}(\boldsymbol{\theta})c^{(t)}(\boldsymbol{\phi})\middle|\middle|\frac{s^{(t)}(\boldsymbol{\theta})c^{(t)}(\boldsymbol{\phi})}{s_i(\boldsymbol{\theta})c_i(\boldsymbol{\phi}_i)}p(\boldsymbol{\theta},\boldsymbol{\phi}_i|\mathcal{D}_i)\right) = \\
= \int d\boldsymbol{\theta}\; s^{(t)}(\boldsymbol{\theta})\log s_i(\boldsymbol{\theta})\int d\boldsymbol{\phi}\; c^{(t)}(\boldsymbol{\phi}) + \int d\boldsymbol{\theta}\; s^{(t)}(\boldsymbol{\theta})\int d\boldsymbol{\phi}\; c^{(t)}(\boldsymbol{\phi})\log c_i(\boldsymbol{\phi}_i) + \\
- \int d\boldsymbol{\theta}\; d\boldsymbol{\phi}\; s^{(t)}(\boldsymbol{\theta})c^{(t)}(\boldsymbol{\phi})\log p(\boldsymbol{\theta},\boldsymbol{\phi}_i|\mathcal{D}_i) = \\
= \int d\boldsymbol{\theta}\; s^{(t)}(\boldsymbol{\theta})\log\frac{s_i(\boldsymbol{\theta})s^{(t)}(\boldsymbol{\theta})}{s^{(t)}(\boldsymbol{\theta})p(\boldsymbol{\theta})} + \int d\boldsymbol{\phi}_i\; c_i^{(t)}(\boldsymbol{\phi}_i)\log\frac{c_i^{(t)}(\boldsymbol{\phi}_i)}{p(\boldsymbol{\phi}_i)} + \\
- \int d\boldsymbol{\theta}\; s^{(t)}(\boldsymbol{\theta})\int d\boldsymbol{\phi}_i\; c_i^{(t)}(\boldsymbol{\phi}_i)\log p(\mathcal{D}_i|\boldsymbol{\theta},\boldsymbol{\phi}_i)
\end{aligned}
$$

where the second equality comes from the normalization of client and server pdfs and from Bayes rule $p(\boldsymbol{\theta},\boldsymbol{\phi}_i|\mathcal{D}_i) \propto p(\mathcal{D}_i|\boldsymbol{\theta},\boldsymbol{\phi}_i)p(\boldsymbol{\phi}_i)p(\boldsymbol{\theta})$. Notice also that $\frac{s_i(\boldsymbol{\theta})}{s^{(t)}(\boldsymbol{\theta})} = \frac{s_i^{(t-1)}(\boldsymbol{\theta})}{s^{(t-1)}(\boldsymbol{\theta})}$ because of the factorization in Equation (2), and hence Equation (3) is proved. □

We can see that the variational free energy in Equation (3) decomposes naturally into two parts. The terms that involve the client parameters $c_i(\boldsymbol{\phi}_i)$ correspond to the variational free energy terms of *Bayes by backprop* [6]. Note that, except for the natural complexity cost given by the second KL term, no additional regularization is applied on the client parameters, that can hence be trained efficiently and network agnostic. The terms that involve the server posterior are instead the likelihood cost and the first KL term. This regularization restricts the server to learn an overall posterior that does not drift from a posterior distribution obtained replacing the current refining factor by the prior. This constraint effectively forces the server to progress in a CL fashion [32], learning from new federated datasets and avoiding catastrophic forgetting of the ones already seen.

The whole free energy in Equation (3) can be optimized using gradient descent and unbiased Monte Carlo estimates of the gradients with reparametrization trick [22]. For simplicity we use a Gaussian mean-field approximation of the posterior, hence for server and client parameters the factorization reads respectively $s_i(\boldsymbol{\theta}) = \prod_{d=1}^{D^s}\mathcal{N}(\theta_d|\mu_{id}^s,\sigma_{id}^s)$ and $c_i(\boldsymbol{\phi}_i) = \prod_{d=1}^{D_i^c}\mathcal{N}(\phi_{id}|\mu_{id}^c,\sigma_{id}^c)$, where D^s and $\{D_i^c\}_{i=0}^K$ are the total number of parameters of the server and of the client networks. A depiction of the full graphical model of the approximated variational posterior is given in Figure 1b. The pseudo-code of `VIRTUAL` is described in Algorithm 1. Notice that similarly to `FedAvg`, privacy is preserved since at any time the server get access only to the overall posterior distribution $s(\boldsymbol{\theta})$ and never to the individual factor $s_i(\boldsymbol{\theta})$ and $c_i(\boldsymbol{\theta})$, that are visible only to the respective client.

Algorithm 1 `VIRTUAL`

Input: datasets $\{\mathcal{D}_1,\dots,\mathcal{D}_k\}$, N number of total refinements, priors $p(\boldsymbol{\theta}), \{p(\boldsymbol{\phi}_i)\}_{i=1}^K$

1: initialize all the pdfs $c_i^{(0)}(\boldsymbol{\phi}_i)$ and $s_i^{(0)}(\boldsymbol{\theta})$
2: $s^{(0)}(\boldsymbol{\theta}) \leftarrow \prod_i s_i^{(0)}(\boldsymbol{\theta})$
3: **for** $t = 1, 2\dots, N$ **do**
4: choose a client i to be refined
5: client computes the new server prior $p(\boldsymbol{\theta})\frac{s^{(t-1)}(\boldsymbol{\theta})}{s_i^{(t-1)}(\boldsymbol{\theta})}$
6: $s_i^{(t)}(\boldsymbol{\theta}), c_i^{(t)}(\boldsymbol{\phi}_i) \leftarrow$ joint optimization of the variational free energy in eq. (3) on client i
7: client computes the new server posterior $s^{(t)}(\boldsymbol{\theta}) \leftarrow \frac{s^{(t-1)}(\boldsymbol{\theta})}{s_i^{(t-1)}(\boldsymbol{\theta})}s_i^{(t)}(\boldsymbol{\theta})$
8: client sends $s^{(t)}(\boldsymbol{\theta})$ to the server
9: server sends its new posterior $s^{(t)}(\boldsymbol{\theta})$ to all clients.

We can further notice an interesting similarity of the `VIRTUAL` algorithm to the *Progress&Compress* method for CL introduced in [38], where a similar free energy is obtained heuristically by composing CL regularization terms and distillation cost functions [18].

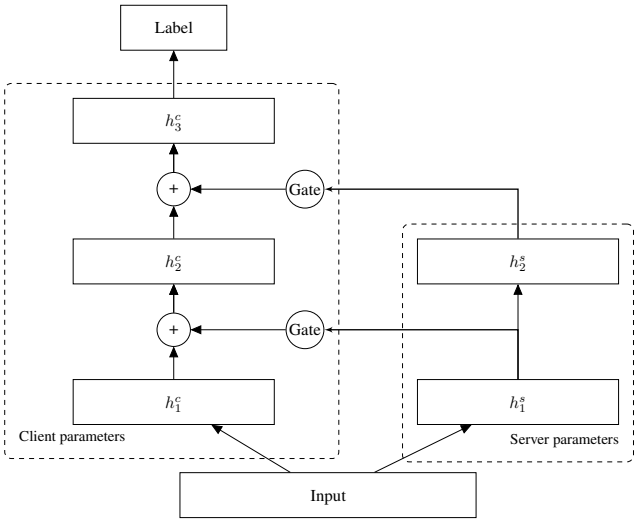


Figure 2: Depiction of the a Server-Client multi-layer perceptron model with two hidden layers and lateral connections. The server parameters on the right plate are shared between all the clients, while the client parameters on the left plate are private. See text for full details.

2.3 The server-client architecture

The model for $p(\mathcal{D}_i|\boldsymbol{\theta}, \boldsymbol{\phi}_i)$ has to be chosen in order to allow the client model to reuse knowledge extracted from the server and enlarging the effective sample size available for client i. In this work we use a model inspired by *Progressive Networks* [36] that has been proved effective in CL, with lateral connections between server and client layers. Lateral connections feed the activations of the server at a given layer as an additive input to the client layer. Formally, for a multi-layer perceptron (MLP) model we can denote by h^s_{l-1} the activation of the server layer at a depth $l-1$, then the subsequent activation of client i, h^c_{il} reads

$$h^c_{il} = \sigma\left(\boldsymbol{\phi}^w_{il} h^c_{l-1} + \boldsymbol{\phi}^\alpha_{il} \odot \boldsymbol{\phi}^u_{il} \sigma(h^s_{l-1} + \boldsymbol{\phi}^b_{il}) + \boldsymbol{\phi}^c_{il}\right) \tag{4}$$

where $\boldsymbol{\phi}^w_{il}, \boldsymbol{\phi}^u_{il}$ are weight matrices, $\boldsymbol{\phi}^\alpha_{il}$ is a weight vector that act as a gate, $\odot$ denotes the element-wise multiplication operator and $\boldsymbol{\phi}^b_{il}$ and $\boldsymbol{\phi}^c_{il}$ are trainable biases. The client-server connection architecture for a 2 hidden-layers MLP is depicted in Figure 2. Note as an example that for the architecture used in Figure 2, the client parameter vector reads $\boldsymbol{\phi}_i = \{\boldsymbol{\phi}^w_{il}, \boldsymbol{\phi}^u_{il}, \boldsymbol{\phi}^\alpha_{il}, \boldsymbol{\phi}^b_{il}, \boldsymbol{\phi}^c_{il}\}_{l=1}^3$.

3 Related Work

We now provide a brief survey of work in the area of distributed/federated learning and of transfer/continuous learning, in light of the problem at hand described in Section 1 and of the tools used in deriving `VIRTUAL`.

Distributed and federated Learning Distributed learning is a learning paradigm for which the optimization of a generic model is distributed in a parallel computing environment with centralized data [29]. Early work on this paradigm propose various learning strategies that require iterative averaging of locally trained models, typically using Stochastic Gradient Descent (`SGD`) steps in the local optimization routine [29, 34, 49, 12]. These works typically consider the distributed learning to be set in a computational cluster, so with few computing devices, fast and reliable communication between devices and centralized unbalanced datasets. FL [30] eliminates all these constraints and it is framed as a paradigm that encompasses the new challenges and desiderata listed in Section 1. `FedAvg` [30, 24] has been proposed as straightforward heuristic for the FL. At each step of the algorithm a subset of the online clients is selected, and these are then updated locally using `SGD`. The models are then averaged to form the model at the next step, which is maintained in the server and transmitted back to all the clients. Despite working well in practice, it has been shown that the performance of `FedAvg` can degrade significantly for skewed non-IID data [30, 50].

Table 1: Statistics of the datasets used in the experiments.

Dataset	Number of clients	Number of classes	Total samples	Samples per client	
				mean	std
MNIST	10	10	60000	6000	0
P-MNIST	10	10	60000	6000	0
FEMNIST	10	62	5560	556	54
VSN	23	2	68532	3115	559
HAR	30	6	15762	543	56

Some heuristics have been proposed recently to solve the statistical challenges of FL. In particular recently it has been proposed to share part of the client-generated data [50] or a server-trained generative model [21] to the whole network of clients. These solutions are however questionable since they require significant communication effort, and do not comply with the standard privacy requirements of FL. Another solution for this problem has been proposed in [37] where the authors extend `FedAvg` into `FedProx`, an algorithm that prescribes clients to optimize the local loss function, further regularized with an quadratic penalty anchored on the weights of the previous step. Despite showing improvements on the `FedAvg` algorithm for very data heterogeneous settings, the method is strongly inspired by early works on continuous and transfer learning (see e.g. see for example Elastic Weight consolidation (EWC) in [23, 48] and the literature review in the next paragraph) and hence can be further refined.

The first contribution to highlight the possibility of naturally embedding FL in the MTL framework has been reported by `MOCHA` [40], that extends some early work on distributed MTL-like `CoCoA` and variations [39, 20, 28]. In this work a federated primal-dual optimization algorithm is derived for *convex* models with MTL regularization, and it is shown for the first time that the MTL framework can enhance the model performance, with the MTL model outperforming global models (trained with centralized data) and local models as well, on real world federated datasets.

Transfer and continuous learning The transfer of knowledge in neural network, from one task to another, has been used extensively and with great success since the pioneering work in [17] of transferring information from a generative to a discriminative model using fine tuning. The application of this straightforward procedure is however difficult to apply in scenarios where multiple tasks from which to transfer from are available. Indeed a good target performance can be obtained only with a priori knowledge of task similarity, that is usually not known, while learning of sequential tasks causes knowledge of previous tasks to be abruptly erased from the network in what has been called *catastrophic forgetting* [15].

Many methods have been introduced to overcome catastrophic forgetting, and to enable models to learn multiple task sequentially retaining a good overall performance, and transferring effectively to new tasks. Many early works proposed different regularization terms of the loss function anchored to the previous solution in order to get new solutions that generalize well on old tasks [23, 48]. These methods have been first introduced as heuristics, but have been found to be applications of well-known inference algorithms like *Laplace Propagation* [41] and *Streaming Variational Bayes* [7], which led to further generalizations [26, 16]. New approaches focused on other components, like architecture innovations, introducing lateral connections that allow new models to reuse knowledge from previously trained models with *layer-wise adaptors* [36, 38], and memory enhanced models with generative networks [45, 47]. A recently introduced online Bayesian inference approach [32] served as inspiration for our work. It frames the continual learning paradigm in the Bayesian inference framework, establishing a posterior distribution over network parameters that is updated for any new task in light of the new likelihood function. It has been showed that this method outperformed all previously known method for CL.

4 Experiments

In this section we present an empirical evaluation of the performance of `VIRTUAL` on several real world federated datasets.

4.1 Dataset description

MNIST: The classic MNIST dataset [25], randomly split into 10 different sections. Note that this dataset has not been generated in a real federated setting having IID client samples.

Permuted MNIST (P-MNIST): The MNIST dataset is randomly split into 10 sections, and a random permutation of the pixels is applied on every section. First used in [48] in the context of CL.

FEMNIST: This dataset consists of a federated version of the EMNIST dataset [10], maintained by the *LEAF* project [9] Different clients correspond to different writers. We sub-sample 10 random writers from those with at least 300 samples.

Vehicle Sensors Network (VSN)[1]: A network of 23 different sensors (including seismic, acoustic and passive infra-red sensors) are place around a road segment in order to classify vehicles driving through. [14]. The raw signal is featurized in the original paper into 50 acoustic and 50 seismic features. We consider every sensor as a client and perform binary classification of assault amphibious vehicles and dragon wagon vehicles.

Human Activity Recognition (HAR)[2]: Recordings of 30 subjects performing daily activities are collected using a waist-mounted smart-phone with inertial sensors. The raw signal is divided into windows and featurized into a 561-length vector [3]. Every individual correspond to a different client and we perform classification of 6 different activities (e.g. sitting, walking).

A comprehensive description of the statistics of the datasets used is available in Table 1.

4.2 Experiment setting

All networks employed are multilayer perceptrons (MLP) with two hidden dense flipout layers [44] with 100 units and ReLU activation functions. Dropout with parameter 0.3 is used before every dense layer. The Monte Carlo estimate of the gradient is performed in all the experiment using 20 samples. In all the experiments, `VIRTUAL` has been evaluated training the clients in an incremental fashion, in fixed order, with 3 refinements per client.

Using the notation of the original paper [30], `FedAvg` has been evaluated in all experiments with a fraction of updated clients per round $C = 0.2$ and a number of epochs per round $E = 1$, that has been show to guarantee converge in all scenarios [30, 9]. The total number of rounds is chosen such that every client is trained on average for a number of epochs that is equal to that of the `VIRTUAL` experiments.

We further compare our algorithm with *Local* and *Global* baselines that are obtained respectively training one separate model per client, and training one single model on centralized data. Global does not comply with the generic federated setting and is reported only as a comparison.

Implementation of `VIRTUAL` is based on *tensorflow* [1] and *tensorflow distributions* [13] packages. [3] The implementation of `FedAvg` is taken from the *Leaf* benchmark for federated learning settings [9].

4.3 Results

In Table 2 we show the advantages given by the multi-task learning framework in the federated setting. In the table we measure the average categorical accuracy over all tasks of the respective dataset. Every experiment has been repeated over 5 random 25% train test splits, and we report mean and standard deviation over the runs.

We can see that the performance of all the algorithms strongly depend on the degree of heterogeneity of the dataset considered. In particular the *Global* baseline is among the top performing methods only on the MNIST dataset, that has been generated in IID fashion. Strongly non-IID scenarios are depicted by the P-MNIST and VSN datasets, that have significantly dissimilar feature spaces among clients (P-MNIST features are given by random permutations, while VSN encompasses a

[1] `http://www.ecs.umass.edu/~mduarte/Software.html`

[2] `https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones`

[3] Code and full details on the hyper-parameter used are available at `https://github.com/lucori/virtual`

Table 2: Multi-task average test accuracy. Mean and standard deviation over 5 train-test splits.

Method	MNIST*	FEMNIST	P-MNIST	VSN	HAR
*Global**	**0.9678±0.0007**	0.44±0.02	0.919±0.004	0.926±0.005	0.797±0.006
Local	0.9511±0.0020	0.51±0.01	**0.950±0.003**	**0.960±0.003**	0.940±0.001
FedAvg	**0.9675±0.0004**	0.45±0.05	0.905±0.002	0.916±0.007	**0.944±0.004**
Virtual	**0.9666±0.0017**	**0.56±0.01**	**0.949±0.001**	**0.960±0.002**	**0.944±0.001**

wide spectrum of different sensors). In these scenarios the performance of both *Global* and FedAvg degrades while *Local* models enjoy high performances, being tuned to the specific data distribution.

We can see that VIRTUAL maintain the top performance in the whole sprectrum of federated scenarios, being on par with *Global* and FedAvg on IID datasets, and with *Local* models on strongly non-IID settings. It also outperforms other methods on FEMNIST and HAR, that are datasets that best represent the multi-task learning setting, as they encompass different users gathering data in very similar but distinct conditions.

5 Conclusion

In this work we introduced VIRTUAL, an algorithm for federated learning that tackles the well known statistical challenges of the federated learning framework using a multi-task setting. We consider the federation of central server and clients as a Bayesian network and perform training using approximated variational inference. The algorithm naturally comply with the federated setting desiderata, giving access to the central server only to an aggregated parameter update in the form of an overall posterior distribution over shared parameters. The algorithm is shown to outperform the state-of-the-art in non-IID real world federated datasets, and to be on par with the state-of-the-art in other scenarios.

One possible direction for further developments is to consider synchronous updates of multiple clients studying empirically the effect of using outdated priors during client training or theoretically developing a new Bayesian model of synchronous updates. Another interesting direction is the exploration of other design choices. Indeed the general method can be tuned for a particular application by modifying e.g. the architecture of the lateral connections between devices (Block-Modular NN [42], NinN architecture [27]), the topology of the Bayesian network (star shape, hierarchical etc.), the choice of the variational inference algorithm. Finally, it is possible to study VIRTUAL under memory constraints, for which on optimal strategy can store chunks of data for further refinements or discard them, in the line of coresets theory [4].

References

[1] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Muhammad Ammad-ud-din et al. "Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System". In: *arXiv preprint arXiv:1901.09888* (2019).

[3] D Anguita et al. "A Public Domain Dataset for Human Activity Recognition using Smartphones". In: *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. CIACO. 2013, pp. 437–442.

[4] Olivier Bachem, Mario Lucic, and Andreas Krause. "Coresets for Nonparametric Estimation-the Case of DP-Means." In: *ICML*. 2015, pp. 209–217.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] Charles Blundell et al. "Weight Uncertainty in Neural Network". In: *International Conference on Machine Learning*. 2015, pp. 1613–1622.

[7] Tamara Broderick et al. "Streaming variational bayes". In: *Advances in Neural Information Processing Systems*. 2013, pp. 1727–1735.

[8] Thang D Bui et al. "Partitioned Variational Inference: A unified framework encompassing federated and continual learning". In: *arXiv preprint arXiv:1811.11206* (2018).

[9] Sebastian Caldas et al. “LEAF: A Benchmark for Federated Settings”. In: *arXiv preprint arXiv:1812.01097* (2018).

[10] Gregory Cohen et al. “EMNIST: an extension of MNIST to handwritten letters”. In: *arXiv preprint arXiv:1702.05373* (2017).

[11] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[12] Jeffrey Dean et al. “Large scale distributed deep networks”. In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[13] Joshua V Dillon et al. “Tensorflow distributions”. In: *arXiv preprint arXiv:1711.10604* (2017).

[14] Marco F Duarte and Yu Hen Hu. “Vehicle classification in distributed sensor networks”. In: *Journal of Parallel and Distributed Computing* 64.7 (2004), pp. 826–838.

[15] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[16] Robin C Geyer, Viktor Wegmayr, and Luca Corinzia. “Transfer Learning by Adaptive Merging of Multiple Models”. In: (2018).

[17] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[19] White House. “Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy”. In: *White House, Washington, DC* (2012), pp. 1–62.

[20] Martin Jaggi et al. “Communication-efficient distributed dual coordinate ascent”. In: *Advances in neural information processing systems*. 2014, pp. 3068–3076.

[21] Eunjeong Jeong et al. “Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data”. In: *arXiv preprint arXiv:1811.11479* (2018).

[22] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[23] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[24] Jakub Konečny et al. “Federated optimization: Distributed machine learning for on-device intelligence”. In: *arXiv preprint arXiv:1610.02527* (2016).

[25] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[26] Sang-Woo Lee et al. “Overcoming catastrophic forgetting by incremental moment matching”. In: *Advances in neural information processing systems*. 2017, pp. 4652–4662.

[27] Min Lin, Qiang Chen, and Shuicheng Yan. “Network in network”. In: *arXiv preprint arXiv:1312.4400* (2013).

[28] Chenxin Ma et al. “Adding vs. averaging in distributed primal-dual optimization”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*. JMLR. org. 2015, pp. 1973–1982.

[29] Ryan McDonald, Keith Hall, and Gideon Mann. “Distributed training strategies for the structured perceptron”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2010, pp. 456–464.

[30] Brendan McMahan et al. “Communication-Efficient Learning of Deep Networks from Decentralized Data”. In: *Artificial Intelligence and Statistics*. 2017, pp. 1273–1282.

[31] Thomas P Minka. “Expectation propagation for approximate Bayesian inference”. In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 2001, pp. 362–369.

[32] Cuong V Nguyen et al. “Variational continual learning”. In: *arXiv preprint arXiv:1710.10628* (2017).

[33] Adrian Nilsson et al. “A performance evaluation of federated learning algorithms”. In: *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning (New York, NY, USA, 2018), DIDL*. Vol. 18. 2018, pp. 1–8.

[34] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. “Parallel training of deep neural networks with natural gradient and parameter averaging”. In: *arXiv preprint arXiv:1410.7455* (2014).

[35] Sebastian Ruder. “An overview of multi-task learning in deep neural networks”. In: *arXiv preprint arXiv:1706.05098* (2017).

[36] Andrei A Rusu et al. “Progressive neural networks”. In: *arXiv preprint arXiv:1606.04671* (2016).

[37] Anit Kumar Sahu et al. “On the Convergence of Federated Optimization in Heterogeneous Networks”. In: *arXiv preprint arXiv:1812.06127* (2018).

[38] Jonathan Schwarz et al. “Progress & Compress: A scalable framework for continual learning”. In: *International Conference on Machine Learning*. 2018, pp. 4535–4544.

[39] Virginia Smith et al. “Cocoa: A general framework for communication-efficient distributed optimization”. In: *Journal of Machine Learning Research* 18 (2018), p. 230.

[40] Virginia Smith et al. “Federated multi-task learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4424–4434.

[41] Alexander J Smola, Vishy Vishwanathan, and Eleazar Eskin. “Laplace Propagation.” In: *NIPS*. 2003, pp. 441–448.

[42] Alexander V Terekhov, Guglielmo Montone, and J Kevin O’Regan. “Knowledge transfer in deep block-modular neural networks”. In: *Conference on Biomimetic and Biohybrid Systems*. Springer. 2015, pp. 268–279.

[43] Paul Voigt and Axel Von dem Bussche. “The EU General Data Protection Regulation (GDPR)”. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* (2017).

[44] Yeming Wen et al. “Flipout: Efficient pseudo-independent weight perturbations on mini-batches”. In: *arXiv preprint arXiv:1803.04386* (2018).

[45] Chenshen Wu et al. “Memory Replay GANs: learning to generate images from new categories without forgetting”. In: *arXiv preprint arXiv:1809.02058* (2018).

[46] Timothy Yang et al. “Applied federated learning: Improving google keyboard query suggestions”. In: *arXiv preprint arXiv:1812.02903* (2018).

[47] Jaehong Yoon et al. “ORACLE: Order Robust Adaptive Continual LEarning”. In: *arXiv preprint arXiv:1902.09432* (2019).

[48] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3987–3995.

[49] Sixin Zhang, Anna E Choromanska, and Yann LeCun. “Deep learning with elastic averaging SGD”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 685–693.

[50] Yue Zhao et al. “Federated learning with non-iid data”. In: *arXiv preprint arXiv:1806.00582* (2018).