# JoVe-FL: A Joint-Embedding Vertical Federated Learning Framework

Maria Hartmann[1][a], Grégoire Danoy[1,2][b], Mohammed Alswaitti[1][c] and Pascal Bouvry[1,2][d]

[1]*SnT, University of Luxembourg, Luxembourg, Esch-sur-Alzette, Luxembourg*
[2]*FSTM/DCS, University of Luxembourg, Esch-sur-Alzette, Luxembourg*

Keywords:     Machine Learning, Distributed Machine Learning, Federated Learning, Vertical Federated Learning.

Abstract:     Federated learning is a particular type of distributed machine learning, designed to permit the joint training of a single machine learning model by multiple participants that each possess a local dataset. A characteristic feature of federated learning strategies is the avoidance of any disclosure of client data to other participants of the learning scheme. While a wealth of well-performing solutions for different scenarios exists for Horizontal Federated Learning (HFL), to date little attention has been devoted to Vertical Federated Learning (VFL). Existing approaches are limited to narrow application scenarios where few clients participate, privacy is a main concern and the vertical distribution of client data is well-understood. In this article, we first argue that VFL is naturally applicable to another, much broader application context where sharing of data is mainly limited by technological instead of privacy constraints, such as in sensor networks or satellite swarms. A VFL scheme applied to such a setting could unlock previously inaccessible on-device machine learning potential. We then propose the Joint-embedding Vertical Federated Learning framework (JoVe-FL), a first VFL framework designed for such settings. JoVe-FL is based on the idea of transforming the vertical federated learning problem to a horizontal one by learning a joint embedding space, allowing us to leverage existing HFL solutions. Finally, we empirically demonstrate the feasibility of the approach on instances consisting of different partitionings of the CIFAR10 dataset.

## 1 INTRODUCTION

Federated learning is a co-operative machine learning strategy that allows distributed devices to compute a joint machine learning model, combining locally available information, without actually sharing their individual data sets with each other. This is generally accomplished by training local machine learning models on the separate data sets owned by each participant and intermittently sharing information about the resulting models (most commonly weights of neural networks) with other participants. These local models can be aggregated mathematically to obtain a more accurate global model. There has been widespread interest in the field of federated learning since it was first proposed by McMahan et al. in 2016 (McMahan et al., 2016), but the overwhelming majority of research contributions (Kairouz et al., 2019) have focused on improving applications to the class of scenarios for which the framework was first de-

signed: performing joint machine learning in a context where participants' datasets may not be transmitted for reasons such as privacy and confidentiality of client information. Indeed, this focus is reflected in a commonly accepted definition of the federated learning concept (Yang et al., 2019) : "A federated learning system is a learning process in which the data owners collaboratively train a model $M_{FED}$ , in which process any data owner $F_i$ does not expose its data $D_i$ to others".

However, the same solution approach lends itself naturally to another type of application scenarios: those where data *cannot* be transmitted for external reasons like technological constraints. Examples of such applications include on-line learning for an autonomous swarm of unmanned aerial vehicles (UAVs) (Brik et al., 2020) carrying out a survey as part of a search-and-rescue mission. In such a context UAVs must limit the amount of information they transmit in order to conserve battery charge and maximise flying range. Using federated learning, a swarm of such devices could train a simple machine learning model predicting the efficiency of particular flying manoeuvres in the specific, potentially difficult, conditions it

[a] https://orcid.org/0000-0002-2179-3703
[b] https://orcid.org/0000-0001-9419-4210
[c] https://orcid.org/0000-0003-0580-6954
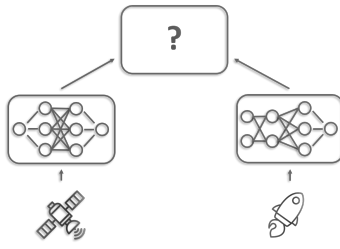[d] https://orcid.org/0000-0001-9338-2834

Figure 1: The central challenge of vertical federated learning is the aggregation of client models with heterogeneous architectures trained on different types of data.

is deployed in. In another example, a constellation of Earth observation satellites in different orbits that pass each other only intermittently in communication range could exchange model information during those short intervals to train a federated machine learning model assisting in simple autonomous navigation procedures such as self-correcting for tumbling motions.

Leveraging Federated Learning in such contexts could enable joint machine learning training on each device even across severely limited communication channels, allowing participants to share information much more efficiently. Learning in a way that maintains a current model on each agent (i.e. device such as a UAV) means that swarming agents may still operate successfully in unstable communications environments, exiting and rejoining the learning process without interruption to the wider system. Furthermore, federated learning can conceivably be performed across networks where not all agents are fully connected, e.g. in large-scale or ad-hoc networks, by propagating what is learned across the network.

Such types of application scenarios call for federated learning solutions with a focus on different challenges that have been addressed less in existing research - this includes e.g. handling unstable connections, limited computational (server) resources or online learning. By far the least studied challenge, however, concerns the integration of multiple different input data and models into one such federated system. The type of applications discussed here extends naturally to include sets of different types of devices with different capabilities, such as UAVs each carrying different sensor equipment.

The pool of existing solutions to this type of federated learning problem (known as vertical federated learning or VFL; see Figure 1 for an illustration) is in general quite sparse, and existing research cannot easily be applied to this context, as many VFL approaches are designed for narrow use cases (Feng and Yu, 2020) or require extensive pre-processing and preliminary exchange of information (Liu et al., 2020). Others propose a more centralised approach

that produces a global model only on a central server (Chen et al., 2020).

We present here a federated learning framework that is designed to address the type of application scenario where participants are sufficiently limited in computational and communication capacity that classical machine learning is not an attractive option. In particular, our framework has been developed to allow participants observing different types of data to train heterogeneous machine learning models while still exchanging information with each other in a federated manner.

The remainder of this article is structured as follows: we first recapture and discuss related work in Section 2, then in Section 3 introduce the JoVe-FL framework in detail. We present a first experimental demonstration of the framework in Section 4. Section 5 contains our conclusions and perspectives.

## 2 RELATED WORK

On a high level, federated learning schemes are commonly classified as *horizontal federated learning (HFL)*, *vertical federated learning (VFL)*, or *federated transfer learning (FTL)* based on the distribution of client data sets (Yang et al., 2019). According to a strict definition of these terms, clients in HFL models share the same feature space, but data is partitioned between clients in the sample space (it may be unevenly distributed among clients, i.e. non-iid). In VFL, clients are partitioned in the feature space, but share the same sample space. When clients share neither the feature space nor the sample space, this is referred to as Federated Transfer Learning (FTL). In practical applications, these categories may be less clear-cut, particularly with respect to the boundary between VFL and FTL (Saha and Ahmad, 2020). Clients are likely to overlap in both the feature space and the sample space to some degree, and in this case the classification of the problem may depend on the size of the overlap or the target of the particular learning problem.

Much progress has been made in various directions in the classical HFL scenario, where all clients possess samples from the same feature space and train the same type of model (Kairouz et al., 2019). Less work exists on the VFL and FTL scenario (Kairouz et al., 2019), where the feature space is subdivided among clients and clients may thus train different model architectures. Of the existing solutions, most strictly consider either the case where clients' feature spaces are guaranteed to overlap (Feng and Yu, 2020), (Zhang and Jiang, 2022) – e.g. each client possesses

part of an image, – or the case where the feature space is fully partitioned. In some cases, it is additionally assumed that labels are also private to a subset of participants, e.g. in (Xia et al., 2021). Approaches to the former generally rely on pre-processing to match partial samples; some utilise data augmentation methods to exploit shared features between clients. Examples include an expansion of multi-view learning into the Multi-Participant Multi-View Federated Learning (MMVFL) framework (Feng and Yu, 2020), as well as a method to artificially extend client samples to a common feature space using generative adversarial networks (Zhang and Jiang, 2022). (Liu et al., 2020) consider the notion of asymmetry in federated learning, where some participants have a significantly stronger interest than others in protecting their sample identities during data-matching. The requirement to perform pre-processing or additional model training makes these approaches fairly computationally expensive and thus ill-suited to scenarios that involve restricted computational resources, e.g. edge devices or satellite networks.

For clients with non-overlapping feature spaces, the SplitNN approach (Gupta and Raskar, 2018) to distributed learning has been adapted for the federated scenario, both for horizontal federated learning(Yuan et al., 2020) and for vertical federated learning (Vepakomma et al., 2018), (Cai et al., 2022). The central insight of the SplitNN architecture is to have each client train a separate network model up to a certain layer known as the cut layer. The output of the cut layer – an embedding of the original input data – is then transferred to a joint model that is trained on the collected output of all clients. In existing federated learning adaptations this generally involves concatenation of all client outputs (Ceballos et al., 2020), thereby binding the server to a fixed number of client inputs that may be subject to communication delays. The Virtual Asynchronous Federated Learning (VAFL) framework (Chen et al., 2020) mitigates such issues by dealing with asynchronicity of client messages.

The Cascade Vertical Federated Learning (CVFL) framework (Xia et al., 2021) extends the concept further to address the asynchronous setting, including the straggler problem, where the label space is also partitioned among clients. CVFL employs bottom-up cascade training, training an embedding model on all participants and an additional top (prediction) model on those participants that possess labels. Each client that owns labels collects the embedding vectors of the clients that do not and concatenates those vectors – as in SplitNN and VAFL – before feeding them into the prediction model. The resulting prediction models are aggregated using a horizontal federated learning strategy. However, another shortcoming of these methods remains: the joint model is incompatible with individual clients. A single client cannot use the results of the joint training, and in return the joint model is only functional while the required number of clients participate reliably.

JoVe-FL overcomes this inflexibility without requiring any concatenation of results. No data matching is required and no constraints are placed on the mutual overlap of client datasets.

# 3 DESCRIPTION OF FRAMEWORK

This section provides a detailed description of the proposed vertical federated learning framework. We consider a scenario in which constraints are dictated primarily by technological limitations instead of privacy concerns. In particular, we make the following assumptions: (1) Each participant knows the labels associated with its own samples. We assume that clients are capable of independent action and learning, and this requires clients to know the labels to their observations. Furthermore, all clients are aware of the full label space; (2) No information is known about the relation between different clients' datasets, i.e. matching of samples or whether the respective feature spaces overlap; (3) Clients do not share raw data with each other or a server; (4) No server with significantly more computing power than any client is available.

## 3.1 Application Scenario

As the framework presented in this paper is intended to solve a class of application scenarios that differs fundamentally from those considered in previous research, it appears worthwhile to first outline these differences. We shall do so with the aid of two examples.

For a classical example of vertical federated learning scenarios, consider the case of several hospitals located in the same geographical region. Each hospital maintains confidential patient files. Different hospitals may have different specialisations, so may be visited by the same patient for different medical needs and thus collect different types of data, such as the results of blood or imaging tests, differential diagnoses or the duration and outcome of hospital admissions. These hospitals could perform machine learning on these data sets to better predict patient outcomes based on symptoms and test results, preferably on the combined data of all regional hospitals

since a larger and broader data set would give a more precise model. However, due to the confidential nature of health data, pooling patient data into one large data set accessible to all participants is not a feasible option. This is a typical example of a scenario that can be solved using federated learning. Using VFL solutions, hospitals can jointly train a global model that includes all information possessed by any one hospital without disclosing patient data. In this scenario, a federated learning scheme typically involves few (probably much less than 100) participants that consolidate large data sets (also known as data silos). There are stable communication channels between participants with few limitations on transmission size or frequency, and little constraint on computing power available to clients and servers. The main concern is the privacy and confidentiality of the training data.

As an example of the type of application problem targeted by JoVe-FL, consider a hypothetical constellation of CubeSats – a class of small research satellites – orbiting Mars on a mapping mission. Each satellite is equipped with onboard imaging sensors and observes a slice of the planetary surface. To obtain maximum coverage, the orbits of individual satellites are arranged such that there is little overlap between their respective observation areas. The objective of each satellite is to classify observed geological features. Note that in fixed orbits satellites may each consistently observe only some of the features occurring on the surface, but this is difficult to predict in advance due to the exploratory nature of the mission. Owing to their limited size, CubeSats are limited in the solar charging equipment they can carry, and so all onboard electronics are necessarily subject to strict energy budgeting to extend the lifespan of the satellite (Arnold et al., 2012). These constraints reduce the available downlink transmission bandwidth such that it is infeasible to simply transmit all observations from the satellites to Earth or a more powerful intermediary for processing. However, on-board computing power is also necessarily limited; therefore a strategy for efficient on-board classification is required. This scenario exhibits some markedly different characteristics to the previously discussed hospital setting, most importantly:

- No powerful server is available, hence any aggregation strategy cannot be too computationally taxing to be carried out by a limited participant.

- The distribution of features between participants may be unknown - a federated learning solution cannot rely on exploiting a particular relation between participants' data sets.

- Participants need to be able to use the trained model locally and independently at any time in the learning process, since satellites may temporarily lose connection or experience technical faults unexpectedly and drop out of the learning process. As a consequence, the global model cannot require the concatenated input of multiple participants to function.

- The main objective in applying a federated learning strategy is to enable efficient information sharing. Data confidentiality is a secondary concern, particularly in deep space missions.

In conclusion, this application scenario – and so also this framework we propose as a first solution approach – is fundamentally client-oriented as opposed to the server-oriented scenario in existing VFL approaches, e.g. (Chen et al., 2020). In existing research, the main aim of vertical federated learning is to obtain an accurate centralised model, with the separate clients as participants but not direct beneficiaries of the solution. In the setting considered here, these aims are reversed: we seek to find accurate models for each client, with the server only used as a means of enabling the learning of more accurate client models.

## 3.2 Concept

The fundamental idea of JoVe-FL, illustrated in Figure 2, is to transform a vertical federated learning problem into a horizontal one, as the concept of horizontal federated learning has been far more well-studied and offers a wide variety of well-performing solution approaches. This transformation is accomplished by training individual local models to map in-
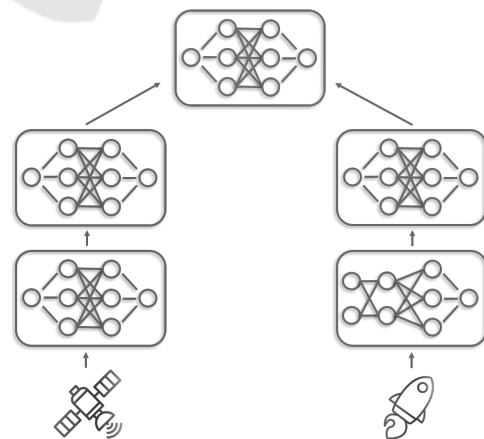


Figure 2: Concept of the framework - Each client maintains a two-level local model. Horizontal federated learning is performed on the top-level models only; the architecture of bottom-level models may be heterogeneous.

put data to a common feature space. On this common feature space a second local model is trained to predict final results. Since the second model operates on a common feature space across all clients, it can be aggregated using a HFL strategy. To the best of our knowledge, JoVe-FL is the first framework to propose handling vertically distributed data implicitly by such a mapping, giving newfound flexibility both regarding client participation and aggregation approaches, and for the first time opening the VFL setting to the existing approaches in the horizontal setting.

## 3.3 Client

Each client locally maintains a model that consists of two chained submodels: an embedding model $\xi_i$ and a prediction model $\theta_i$ as illustrated in Figure 3. The *embedding model* first maps the client's input to an embedding vector of fixed size that is passed to the prediction model. This embedding is then mapped to the final output by the *prediction model*. The choice of model architecture for the embedding model is free for each individual client, with the exception of some constraints on output size that will be detailed below. The architecture of the prediction model is fixed across all clients.

Consider $m$ clients $C_1, \ldots, C_m$. Each client $C_i$ possesses a dataset $X_i$ with samples $x_{i,j}, |x_{i,j}| = n_i$ and associated labels $Y_i$, where $Y_i$ consists of elements of the full label space $Y$ that is known to all clients. Each client defines an embedding model $\xi_i$, with architectures of these models bound only by the constraint that the respective output tensors must have the same dimensions:

$$\forall i \forall j \; \xi_i(x_{i,j}) =: h_{i,j} \; s.t. \; |h_{i,j}| = k. \tag{1}$$

Furthermore, each client has a prediction model $\theta_i$, whose architecture – unlike that of the embedding models – is fixed across all clients. Concatenated the two local models produce a prediction label $\hat{y}_{i,j}$ for each sample, i.e. $\theta_i \circ \xi_i = \theta_i(h_{i,j}) = \hat{y}_{i,j}$. The models are trained using classical gradient descent, with the loss gradient passed through accordingly to the embedding model. Locally, this training is functionally and mathematically equivalent to training a single complete model $\Xi_i := \theta_i \circ \xi_i$. To illustrate the concept, an example instantiation of a JoVe-FL client is shown in Algorithm 1; the corresponding server instantiation follows in Section 3.4. A more detailed discussion of instantiation choices follows in Section 4.2.

## 3.4 Server

The role of the server in JoVe-FL is to aggregate *only the prediction models*, i.e. the top model layer

**Init:** current learning rate $\eta_c$,
local embedding model $\xi_i$
local prediction model $\theta_i$,
last top-1 accuracy $acc_i \leftarrow 0$
**while** *server is active* **do**

    send $\theta_i, acc_i$ to server;
    wait for server update;
    $\theta_0, \eta_c \leftarrow$ receive global model from server;
    $\theta_i \leftarrow \theta_0$;
    $\xi_i, \theta_i \leftarrow$ train $\theta_i \circ \xi_i$ for one epoch;
    $acc_i \leftarrow$ test $\theta_i \circ \xi_i$;
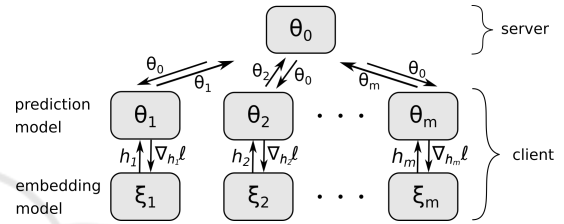**end**

Algorithm 1: JoVe-FL algorithm instance on client $i$.



Figure 3: Formal architecture of the framework - Each client maintains an embedding model $\xi_i$ and a prediction model $\theta_i$. HFL is performed on the prediction models $\theta_i$ to obtain a global model $\theta_0$. The training loss is passed back to the embedding model as in a classical neural network.

$\theta_i$ of each client $C_i$ (see Figure 3). By design of the client architecture, the embedding models are each instances of the same model architecture. We strategically exploit this setup in the design of the server by treating the aggregation problem as equivalent to a horizontal federated learning (HFL) problem. As previously noted, in HFL the clients are assumed to share the same feature space, which is not necessarily true for the input that is fed into the prediction models in this setting. However, by treating the aggregation as if this were the case, we aim to force the clients to compensate by each training the individual bottom-layer embedding models to map their local input to an embedding in a common feature space. JoVe-FL permits considerable freedom of choice regarding server behaviour. By design of the clients, the server can essentially use any aggregation strategy known from classical horizontal federated learning.

Algorithm 2 shows the server instantiation of the framework corresponding to the client intantiation in Algorithm 1.

**Init:** Number of clients $m$,
learning rate $\eta$, current learning rate $\eta_c$
learning rate factor $\varepsilon$
global prediction model $\theta_0$,
patience $p$
approximate global accuracy $acc_g$
**while** $\eta_c > \varepsilon^4 \cdot \eta$ **do**
    Wait for all clients to update;
    $\theta_1, \ldots \theta_m \leftarrow$ read client updates;
    $acc_1, \ldots, acc_m \leftarrow$ read client updates;
    $\theta_0 \leftarrow \text{mean}(\theta_1, \ldots, \theta_m)$;
    $acc_g \leftarrow \min(acc_1, \ldots, acc_m)$;
    $\eta_c \leftarrow$
      recomputeLearningRate$(\eta_c, acc_g, \varepsilon, p)$;
    Send $\theta_0, \eta_c$ to all clients;
**end**
Send termination command to all clients.

Algorithm 2: JoVe-FL algorithm instance on the server.

## 3.5 Full Algorithm

The full federated learning algorithm consists of two alternating phases: the learning phase and the aggregation phase. In this formulation of the framework we describe only the case of synchronous communication between the clients and server; this can doubtlessly be extended in future work to include the asynchronous setting. In the learning phase, each client trains its full individual model on the local dataset. As detailed in section 3.3, this involves sequential application of the embedding model and the prediction model to generate a label prediction. Both models are trained at once using stochastic batch gradient descent on the local labels. This local training phase continues for a given duration, determined by the precise instantiation of the framework. Possible termination conditions include e.g. passage of a given time threshold on the server side or completion of a given number of training iterations or epochs on all clients. The aggregation phase is initiated by the server once a training phase has concluded. During this phase, local clients pause their training process – independently or upon request of the server – and the server collects the current prediction models of participating clients. A HFL aggregation strategy is performed on the collected prediction models, and gradients are passed back to the clients to update the local embedding models depending on the choice of aggregation strategy. Once this is completed, the next local learning phase begins. This process continues until a termination condition on the server is satisfied. Choices of termination condition might include all clients reaching a certain prediction accuracy, completion of a predefined number of aggregation rounds or plateauing of the loss function across all clients.

**Init:** Number of clients $m$,
learning rate $\eta$
learning rate factor $\varepsilon$
global prediction model $\theta_0$,
patience $p$
approximate global accuracy $acc_g$
Initialise server $S(m, \eta, \eta, \varepsilon, \theta_0, p, 0)$ **for** $i \leftarrow 1$
 **to** $m$ **do**
    Initialise client $C_i(\eta, \xi_i, \theta_i)$
**end**

Algorithm 3: Full JoVe-FL algorithm instance.

## 4 EXPERIMENTS

Due to the relative novelty of the sub-field of vertical federated learning, there exists, to the best of our knowledge, no VFL algorithm that addresses a comparable setting. The most closely related work has been explored in the eponymous Section 2, but targets very different application scenarios, as discussed in Sections 2 and 3.1. Consequently, different assumptions about data distribution and available resources are made in the design of these algorithms, which makes a direct comparison with JoVe-FL fruitless. Therefore, in line with other works on different VFL scenarios (Chen et al., 2020), (Xia et al., 2021), we compare the performance of JoVe-FL to 1) the performances of individual clients without cooperation as a lower bound and 2) the state-of-the-art results on the centralised dataset as an upper bound, where available. Where the overlap of our self-generated split datasets does not correspond to the full feature space, no state-of-the-art results of the centralised data are readily available. For these experiments we generate upper bounds using the centralised data and the same model as is used on the clients for the federated learning.

For the first demonstration of the joint-embedding vertical federated learning concept proposed in this paper, we perform proof-of-concept experiments on the CIFAR10 dataset (Krizhevsky and Hinton, 2009). This and similar image classification datasets are widely used for benchmarking in similarly fundamental machine learning experiments (Chen et al., 2020), (Feng and Yu, 2020). Other datasets often used in federated learning research include health- and finance-related data, such as the MIMIC-III and Parkinson datasets. As these simulate application scenarios that our approach is explicitly not designed for, we refrain from using them for our demonstrations in the interest of clarity.
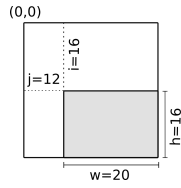
Figure 4: Image split parameters used to generate vertical client inputs. The shaded rectangle corresponds to the parameter selection $i = 16, j = 12, w = 20, h = 16$.

## 4.1 Problem Instances

Experiments were run on different combinations of image parameters: in general, each client observes a slice of fixed size and location of each image, but different sections of the image are revealed to each client. The parameters across different experiments are chosen such that in some cases, client views overlap, in others the image is fully partitioned between clients. These choices have been designed to test the feasibility of JoVe-FL across different types of client data distributions. Defined precisely, the image parameters for each client are defined as follows: $i, j$ are the indices of the top left corner of the rectangular image slice visible to the given client, relative to top left corner of the full image; $w, h$ are the width and height of the visible slice (see Figure 4). We consider five different experimental configurations, shown in Figure 5 and Table 1. We run each configuration five times and average the results across experimental runs.

## 4.2 JoVe-FL Instantiation and Implementation

Experiments were implemented in python using the MIT-licensed OARF benchmarking suite (Hu et al., 2022a),(Hu et al., 2022b), for federated learning systems, which uses pytorch to implement its machine learning aspects. A new server and client module were implemented within the existing code frame-

work. The simulation framework was also modified such that each client may be passed a separate embedding model and individual data splitting parameters upon launch of the simulation.

For the purpose of demonstrating the feasibility of JoVe-FL, we choose the classical synchronous FedAvg algorithm (McMahan et al., 2016) as the aggregation mechanism for the prediction model. The server computes the global prediction model as the true average of all submitted models, and at the beginning of each round all clients replace their local prediction model with the global parameters. This process continues until it is ended by the server. In accordance with the choice of aggregation algorithm, the initial choice of hyperparameters is also modeled on the choices made for the pre-existing implementation of the FedAvg algorithm in OARF. The learning rate is managed on the server side using established pytorch functionalities: a gradual warm-up scheduler increases the initial learning rate linearly from a very small value to the intended starting rate over a small number of epochs. Then, the learning rate is multiplied by a reduction factor during the training whenever the training loss plateaus. After the fourth such reduction, the learning process is ended. Since the server does not possess a complete model and so cannot compute test losses independently, it instead considers the minimum test loss across participating clients.

A sensitivity analysis was carried out to determine the optimal numerical instantiation of model parameters. The parameters that were explored and the resulting selection are listed in Table 2. In total, 432 candidate parameterisations were tested over five experimental runs each, leading to a total of 2160 runs. All experiments were executed on the same type of image input slices with five different random seeds per parameterisation. Computations were performed using the HPC facilities of the University of Luxembourg (Varrette et al., 2022), on multi-GPU nodes with the following hardware specifications per node:

Table 1: Experimental configurations used to generate vertically distributed data for different experiments.

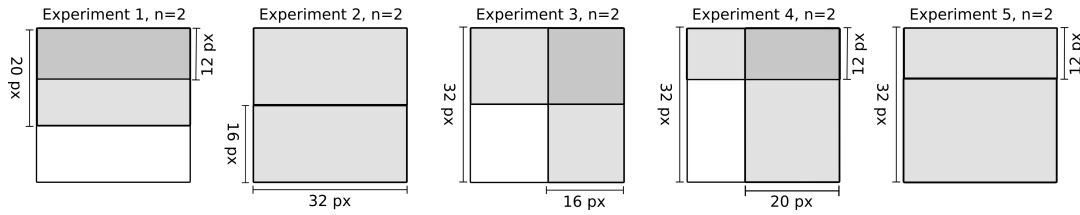| Exp. ID | view | View dimensions | (top, left) idx | # clients |
|---|---|---|---|---|
| 1 | smaller | $12 \times 32$ | $(0,0)$ | 1 |
| | larger | $20 \times 32$ | $(0,0)$ | 1 |
| 2 | equal-size | $16 \times 32$ | $(0,0)$ | 1 |
| | | | $(16,0)$ | 1 |
| 3 | equal-size | $16 \times 32$ | $(0,0)$ | 1 |
| | | $32 \times 16$ | $(0,20)$ | 1 |
| 4 | smaller | $12 \times 32$ | $(0,0)$ | 1 |
| | larger | $32 \times 20$ | $(0,12)$ | 1 |
| 5 | smaller | $12 \times 32$ | $(0,0)$ | 1 |
| | larger | $20 \times 32$ | $(12,0)$ | 1 |

Figure 5: Image split patterns used to generate vertical client inputs. Each shaded rectangle corresponds to one slice of the image visible to an individual client; n denotes the number of clients.

Table 2: Choice of parameters for the experimental model.

| Parameter | Value |
|---|---|
| Learning rate | 0.1 |
| Learning rate reduction factor | 0.2 |
| Batch size | 128 |
| Warmup epochs | 15 |
| Patience | 15 |

Dual Intel Xeon Skylake CPU (28 cores), 4 Nvidia Tesla V100 SXM2 GPU accelerators (16 or 32 GB), 768 GB RAM. On this hardware the experiments required for our sensitivity analysis consumed a total of about 2000 hours of computing time.

In all experiments, each client receives the same number of training and test sample images as all other clients. 83.3% of images are reserved for training and 8.33% are used for testing. Aggregation of the prediction model is performed once per epoch. For simplicity, none of the encryption measures that the OARF framework provides are used in our experiments.

All clients train instances of the same model architecture: the ResNet18 model, split after the second of four block layers (i.e. the third and fourth layer form the prediction model used for the federated aggregation). Locally, each client trains the full ResNet18 model.

As a classical and widely used dataset, the CIFAR10 image dataset for object classification, available under the MIT license, was chosen as the basis for experimental validation. The full image dataset is made available in the OARF implementation, and a modified parameterised pre-processing approach was implemented within the code framework to generate vertically partitioned data. For the partitioned dataset, each client receives a partial view of each image defined by a parameterised rectangle (see Figure 4). The views available to separate clients may overlap fully, partially, or not at all depending on the choice of parameters.

## 4.3 Results and Discussion

Numerical results are presented in Table 3, showing the minimum, average, and maximum top-1 accuracies achieved by clients in JoVe-FL, by clients learning separately on the same split datasets without communication, and by the same model trained on the centralised dataset. In cases where the two clients receive inputs of different sizes (i.e. experiments 1, 4 and 5), the results are presented separately for each of the two clients. In the other cases, results are averaged over both clients. The learning curves for all experiments are plotted in Figures 6, 7 and 8.

Across all experiments, we consistently observe that federated clients obtain a final top-1 accuracy that is close to or better than the one achieved by individual clients without communication. In particular, clients that have a smaller or equally-sized feature space compared to their counterpart perform better in federation than individually across all experiments.

The results indicate that all clients succeed in learning a joint embedding space in all instances, irregardless of whether the data available to clients is fully partitioned or has overlapping sections. Further, this embedding combined with the federated prediction model appears to be effective in many cases at transferring information between clients without requiring a more explicit encoding of information.

In only one experiment (Experiment 2, results shown in Figure 6b) no gains are made by the federated clients compared to the non-federated clients. The input views passed to the clients in this experiment are of equal size and do not overlap; it is possible that in this case the client models converged on an embedding arrangement where the embedding space is simply partitioned between the clients.

In three of our experiments, both clients learn a more accurate classification model using JoVe-FL than they do without communication. In experiment 3, both clients possess image slices of equal size, and both clients benefit from the federated learning - see Figure 7a for the detailed results. In both experiments 4 and 5, the two clients observe image slices of different sizes - overlapping in the case of experiment 4, separate in the case of experiment 5. It appears that an overlap between the participants' datasets is not necessary to find an effective embedding, but neither is it a barrier to information transfer. In-

deed, both clients in experiment 4 make slightly larger accuracy gains compared to the non-communicating clients than those in experiment 5; so it appears possible that the overlap between datasets is recognised while learning a joint embedding and so helps implicitly in aligning the embedding models.

In one case the information gain is unilateral - we observe this in experiment 1, which represents the edge case where one client observes a subset of the feature space that is available to the other client. Since both client datasets originate from the same sample space, we cannot expect any information gain from the client with the larger feature space; but we observe that it appears to lose accuracy in this case. It is possible that in this case attempts at learning a joint embedding space disturbed the learning process of this client and so prevented it from converging to the optimal solution achieved by the equivalent non-federated client. However, we note that the client with the smaller dataset does benefit from the exchange, supporting the fundamental idea of transferring information in this manner. It appears likely that the loss of accuracy on the part of the first client can be avoided by a more careful choice of model architecture; this is an interesting edge case to explore in future work.

Importantly, these preliminary experiments confirm that JoVe-FL handles different vertical distributions of client data – both overlapping and fully partitioned feature spaces – similarly well. To the best of our knowledge, this in itself is a novel characteristic for a VFL framework. In addition we confirm that, even with very little modification to a machine learning model not developed for this purpose and on a basic dataset, JoVe-FL is capable of transferring information between participants with little additional overhead. Nonetheless, the upper accuracy bound computed on the combined datasets of clients is not reached by the federated clients in these preliminary experiments; this requires further development.

## 5 CONCLUSION

This article proposed to explore the potential of applying VFL strategies to settings where machine learning capability is not constrained by privacy concerns, but by other factors, such as technical ones, that also limit participants' ability to share large amounts of data. Different application scenarios have been discussed, particularly in the context of vertical federated learn



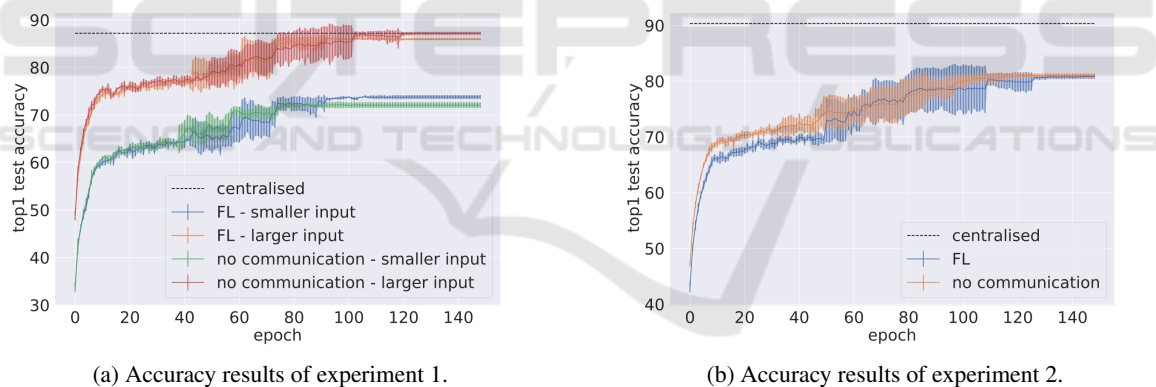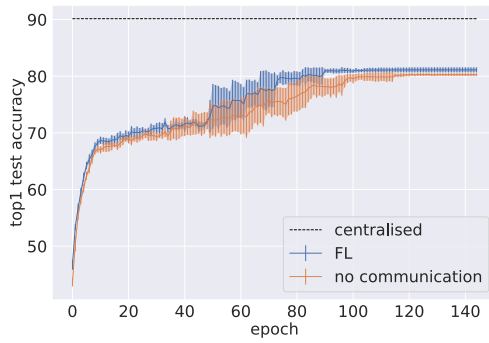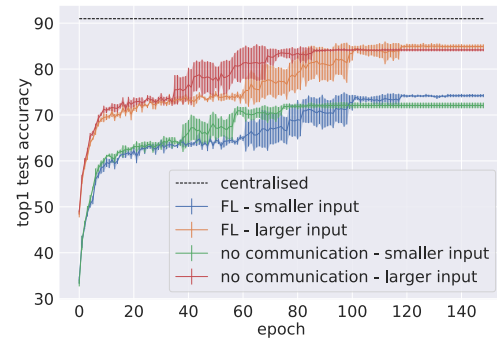(a) Accuracy results of experiment 1.         (b) Accuracy results of experiment 2.

Figure 6: Average top-1 accuracy for unequally-sized views of the dataset with one client dataset a subset of the other (left; Exp. ID 1 in Figure 5) and for equally-sized views of the dataset with no mutual overlap (Exp. ID 2 in Figure 5).

Table 3: Aggregated experimental results of federated learning scheme compared to learning without communication and learning on the centralised dataset.

| Exp. ID | View | FL – top-1<br>min, avg, max | no communication – top-1<br>min, avg, max | centralised – top-1<br>min, avg, max |
|---|---|---|---|---|
| 1 | smaller | **73.0, 73.73, 74.16** | 71.5, 72.1, 73.3 | 86.8, 87.2, 87.6 |
|   | larger | 85.7, 86.0, 86.3 | **86.8, 87.2, 87.6** | |
| 2 | equal-size | 80.2, 80.8, 81.2 | **80.5, 81.0, 81.6** | 90.0, 90.03, 90.9 |
| 3 | equal-size | **80.6, 81.1, 81.91** | 79.9, 80.3, 80.6 | 89.0, 90.1, 90.3 |
| 4 | smaller | **73.7, 74.2, 74.65** | 71.5, 72.1, 73.3 | 90.6, 91.0, 91.3 |
|   | larger | **84.22, 84.9, 85.4** | 83.6, 84.2, 84.5 | |
| 5 | smaller | **72.9, 73.7, 74.0** | 71.5, 72.1, 73.3 | 90.0, 90.3, 90.9 |
|   | larger | **84.5, 85.0, 85.2** | 84.2, 84.5, 85.1 | |

(a) Accuracy results of experiment 3.



(b) Accuracy results of experiment 4.

Figure 7: Average top-1 accuracy for equally-sized views of the dataset with partial mutual overlap (left; Exp. ID 3 in Figure 5) and for unequally-sized views of the dataset with partial mutual overlap (Exp. ID 4 in Figure 5).
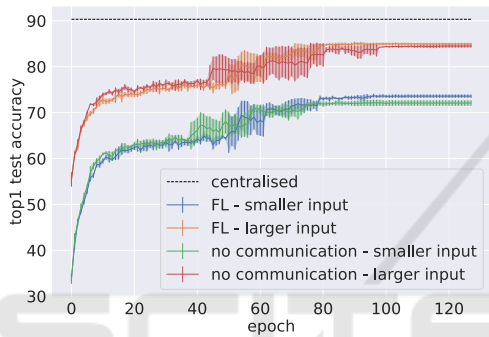


Figure 8: Average top-1 accuracy for unequally-sized views of the dataset with no mutual overlap (Exp. ID 5 in Figure 5).

ing. We propose JoVe-FL, a novel vertical federated learning scheme for flexible learning tailored to this setting. JoVe-FL is, to the best of our knowledge, the first to map a VFL problem to a horizontal one by learning a joint embedding space instead of concatenating separate embeddings. This approach offers remarkable flexibility regarding the partition of data between participants, and it reduces the reliance on consistent participation of clients that is inherent to approaches that use a concatenated embedding space.

We demonstrate the feasibility of our proposed scheme by showing experimentally that clients can implicitly learn a shared embedding space for vertically-distributed data while ultimately achieving a performance accuracy that is at least equal to that of non-communicating clients, and accomplishes superior performance of at least one participating client in all but one of the experiments. Future work will include more extensive benchmarking of JoVe-FL, involving further experimentation on the choice of model architecture both for embedding and prediction models as well for the choice of horizontal aggrega-

tion strategy. In addition, further testing and development towards deploying the framework in application use cases such as swarms of unmanned aerial vehicles or small satellites.

## ACKNOWLEDGEMENTS

## REFERENCES

Arnold, S. S., Nuzzaci, R., and Gordon-Ross, A. (2012). Energy budgeting for cubesats with an integrated fpga. In *2012 IEEE Aerospace Conference*, pages 1–14.

Brik, B., Ksentini, A., and Bouaziz, M. (2020). Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems. *IEEE Access*, 8:53841–53849.

Cai, S., Chai, D., Yang, L., Zhang, J., Jin, Y., Wang, L., Guo, K., and Chen, K. (2022). Secure forward aggregation for vertical federated neural networks.

Ceballos, I., Sharma, V., Mugica, E., Singh, A., Roman, A., Vepakomma, P., and Raskar, R. (2020). SplitNN-driven vertical partitioning. *CoRR*, abs/2008.04137.

Chen, T., Jin, X., Sun, Y., and Yin, W. (2020). VAFL: a method of vertical asynchronous federated learning. *CoRR*, abs/2007.06081.

Feng, S. and Yu, H. (2020). Multi-participant multi-class vertical federated learning. *CoRR*, abs/2001.11154.

Gupta, O. and Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. *CoRR*, abs/1810.06060.

Hu, S., Li, Y., Liu, X., Li, Q., Wu, Z., and He, B. (2022a). The OARF benchmark suite: Characterization and implications for federated learning systems. *ACM Trans. Intell. Syst. Technol.*, 13(4).

Hu, S., Li, Y., Liu, X., Li, Q., Wu, Z., and He, B. (2022b). The OARF benchmark suite: Characterization and implications for federated learning systems [source code], https://github.com/xtra-computing/oarf.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2019). Advances and open problems in federated learning. *CoRR*, abs/1912.04977.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario.

Liu, Y., Zhang, X., and Wang, L. (2020). Asymmetrical vertical federated learning. *CoRR*, abs/2004.07427.

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629.

Saha, S. and Ahmad, T. (2020). Federated transfer learning: concept and applications. *CoRR*, abs/2010.15561.

Varrette, S., Cartiaux, H., Peter, S., Kieffer, E., Valette, T., and Olloh, A. (2022). Management of an Academic HPC & Research Computing Facility: The ULHPC Experience 2.0. In *Proc. of the 6th ACM High Performance Computing and Cluster Technologies Conf. (HPCCT 2022)*, Fuzhou, China. Association for Computing Machinery (ACM).

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. (2018). Split learning for health: Distributed deep learning without sharing raw patient data. *CoRR*, abs/1812.00564.

Xia, W., Li, Y., Zhang, L., Wu, Z., and Yuan, X. (2021). A vertical federated learning framework for horizontally partitioned labels. *CoRR*, abs/2106.10056.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2).

Yuan, B., Ge, S., and Xing, W. (2020). A federated learning framework for healthcare iot devices. *CoRR*, abs/2005.05083.

Zhang, J. and Jiang, Y. (2022). A data augmentation method for vertical federated learning. *Wireless Communications and Mobile Computing*, 2022:1–16.