Vertical Federated Knowledge Transfer via Representation Distillation

Leye Wang[1] , **Chongru Huang**[2] and **Xiao Han**[3]
[1]Peking University, China
[2]Northwest A&F University, China
[3]Shanghai University of Finance and Economics, China

Abstract

With the emergence of data protection laws and regulations, federated learning (FL) has played an important role in cross-party feature enrichment for a machine learning system. Such FL methods on feature enrichment are often known as vertical FL. Traditional vertical FL can only benefit multiple parties' shared samples, which strongly restricts its application scope. To expand vertical FL's usability to each party's (non-shared) local samples, we propose a vertical federated knowledge transfer mechanism based on a novel cross-party representation distillation component. Specifically, our mechanism includes three steps. First, shared samples' *federated representations* are learned by considering multiple parties' joint features with an efficient randomized-masking-based federated matrix decomposition method. Second, for each party, we learn a *federated-representation-distilled auto-encoder*, which can distill the knowledge from shared samples' federated representations to enrich local samples' representations. Finally, each party can leverage local samples' representations enriched by the distilled auto-encoder to boost an arbitrary machine learning task. The experiments on real-life datasets verify the knowledge transfer effectiveness of our mechanism.

1 Introduction

With the prevalence of data protection laws such as GDPR (General Data Protection Regulation), how to conduct machine learning and data mining in a privacy-preserving and law-regulated way has attracted much interest in both academia and industry. Federated learning (FL) has thus become one promising solution [Yang *et al.*, 2019]. In general, FL does not need different parties to exchange their raw data; instead, every party runs local computation and training on their own data and then uploads the intermediate results (e.g., gradients) to a server. By integrating these intermediate results from all the parties, a federated global model can be learned. Especially, such a federated model can achieve similar prediction performance as a centralized model directly trained on all the parties' data [Yang *et al.*, 2019].

In general, there are two main types of FL algorithms, *horizontal* and *vertical*. The first FL algorithm proposed by Google is horizontal [McMahan *et al.*, 2017]; the setting is that different parties (often devices) hold different samples with same features or data formats. A representative application of horizontal FL is the mobile phone keyboard next-word prediction, where a global next-word prediction model can be learned without collecting users' raw keyboard inputs [Yang *et al.*, 2018]. In contrast, vertical FL's setting is that different parties (often organizations) hold different features of the same set of samples. This work focus on the vertical setting.

The successful adoption of current vertical FL methods is highly dependent on how many overlapped samples exist between parties. Hence, most vertical FL collaborations are conducted by involving *at least one giant data holder with abundant data*. For instance, FDN (federated data network)[1] includes anonymous data from one of the largest social network service providers in China and thus can cover most user samples from other data holders (e.g., customers of banks). However, this makes giant data holders occupy a dominant position over other small data holders in vertical FL, which could lead to unfair trades and data monopoly in the digital economy.[2] To alleviate this pitfall and expand application scenarios, *a vertical FL process that can benefit various ordinary data holders (e.g., two parties with a small number of overlapped samples) is urgently needed.*

As a pioneering attempt in this direction, this paper proposes a novel vertical federated knowledge transfer algorithm that can transfer the knowledge from (a limited number of) cross-party shared samples to each party's local (non-shared) samples. The key challenge is, *how to fill the gap between a party's local samples (with only this party's features) and cross-party shared samples (with multiple parties' features).* To address this issue, we propose a novel distilled auto-encoder, which can distill the knowledge from shared samples' federated representations to enrich local samples' representations. More specifically, shared samples' federated representations are first learned by some federated latent representation extraction methods (e.g., federated singular vector decomposition [Chai *et al.*, 2021]); then, a party can leverage

[1]https://fdn.webank.com/

[2]https://www.theguardian.com/technology/2015/apr/19/google-dominates-search-real-problem-monopoly-data

shared samples' federated representation as the guidance to enrich its local feature auto-encoder via a knowledge distilling strategy [Hinton *et al.*, 2015]. Especially, our knowledge transfer mechanism has the following characteristics.

- *Knowledge transfer to local samples*. As aforementioned, different from most vertical FL algorithms focusing on shared samples, our mechanism aims to improve the learning performance on different parties' local samples via vertical knowledge transfer. In this way, a set of parties with only a limited number of shared samples can still benefit from our vertical FL process.
- *Task-independent transfer*. Our knowledge transfer process is task-independent. That is, each party can leverage their enriched local samples' representations for an arbitrary (new) learning task.
- *Salable to multiple parties*. The complexity of our mechanism is linearly proportional to the number of parties. More importantly, our mechanism can be learned in an online manner. That is, when a new party comes, existing parties can efficiently update their local sample representations by just learning with the new party.

In summary, this work makes the following contributions:

1. To the best of our knowledge, this work is the first one to explore how to enable vertical knowledge transfer from shared samples to each party's local samples in a task-independent manner.
2. We propose a novel *federated-representation-distilled auto-encoder* framework to transfer knowledge from shared samples to local samples. This framework includes the following main steps. First, a federated representation learning method is applied to extract shared samples' representations. Second, each party can enrich their local feature auto-encoder by knowledge distilling on the shared samples' federated representations. The distilled auto-encoder can then be leveraged to enrich local samples' feature representations.
3. Experiments on three real-life datasets have verified the effectiveness of our mechanism in knowledge transfer. Specifically, by varying a spectrum of experimental parameters, we have verified the generalizability of our enriched feature representations of local samples.

2 Problem Formulation

In this section, we clarify the definitions of key concepts used in this paper. Afterward, we formulate our research problem.

2.1 Concepts

For all the parties in a vertical FL campaign, we classify them into two types, the *task party* and the *data party*.

Task Party. A task party t has a set of samples with features X_t and a task label Y_t to predict. The task party sample IDs are denoted as I_t.

Data Party. A data party d has a set of samples with features X_d. The data party d's sample IDs are denoted as I_d.

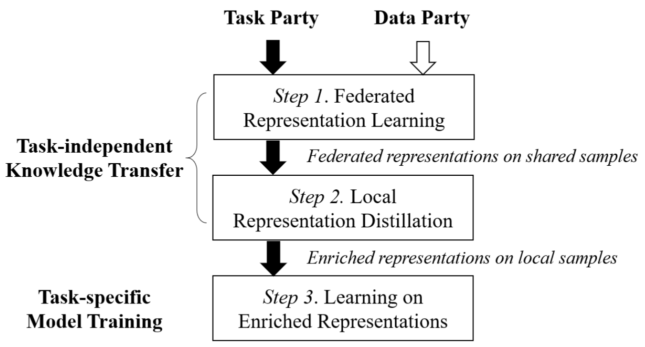
Task Party
Data Party
Step 1. Federated Representation Learning
Task-independent Knowledge Transfer
Federated representations on shared samples
Step 2. Local Representation Distillation
Enriched representations on local samples
Task-specific Model Training
Step 3. Learning on Enriched Representations

Figure 1: Overview of our mechanism.

2.2 Research Problem

Multi-Party Vertical Federated Knowledge Transfer Problem. Given a task party t and n data parties $d_i (i = 1, 2, ..., n)$, t has certain shared samples with any data party d_i ($I_t \cap I_{d_i} \neq \phi$), and t's features are distinct from any data party d_i ($X_t \cap X_{d_i} = \phi$), the objective is to design a federated knowledge transfer algorithm to predict the task label Y_t of t's (non-shared) local samples as accurately as possible.

Remark. Traditional vertical FL problem often requires that $I_t = I_{d_i}$. However, our vertical federated knowledge transfer setting only needs that $I_t \cap I_{d_i} \neq \phi$. The objective of our proposal is to improve the task performance of t's local samples ($I_t \setminus I_{d_i}$) by transferring the knowledge from shared samples ($I_t \cap I_{d_i}$). This is complementary to traditional vertical FL, thus extending FL's application scope in practice.

3 Mechanism Design

3.1 Overview

We demonstrate the overall process of our mechanism, which includes three main steps (Fig. 1). Note that before our mechanism runs, we suppose that shared samples between the task party t and any data party d_i are known, which can be learned by PSI (private set intersection) methods [Kamara *et al.*, 2014].

- **Step 1. Federated Representation Learning**. First, the task party and data parties collaboratively learn federated latent representations for shared samples. In brief, these federated latent representations would incorporate the hidden knowledge among multiple parties, while not leaking these parties' raw features.
- **Step 2. Local Representation Distillation**. Second, the task party trains a *federated-representation-distilled auto-encoder* that can distill the knowledge from shared samples' federated representations to enrich local samples' representations.
- **Step 3. Learning on Enriched Representations**. After Step 2, the auto-encoder is distilled and ready for local feature enrichment. Then, given an arbitrary label y_t to predict, the task party can use local samples' enriched representations (i.e., task party's local features + enriched representations) to conduct training and inference with state-of-the-art machine learning algorithms.

Step 3 generally follows traditional supervised learning methods to train a task-specific prediction model, where various machine learning algorithms can be applied, such as random forest and neural networks. Next, we illustrate more details about Step 1 and 2. For brevity, we first assume that only one data party d exists. Then, we discuss how to deal with multiple data parties $\{d_1, d_2, ..., d_n\}$.

3.2 Federated Representation Learning

The purpose of Step 1 is to extract latent representations of shared samples by considering both task and data parties' features. Literature has shown that singular vector decomposition is effective to extract meaningful latent representations for machine learning tasks [Kosinski *et al.*, 2013]. To this end, based on a federated singular vector decomposition method, FedSVD [Chai *et al.*, 2021], we design a process to learn shared samples' federated representations by considering both task and data parties' features.

Suppose the task party holds the shared samples' feature matrix $S_t \in \mathbb{R}^{|I_s| \times |X_t|}$, and the data party d holds the shared samples' feature matrix $S_d \in \mathbb{R}^{|I_s| \times |X_d|}$ ($I_s = I_t \cap I_d$ is the shared sample ID set). Denote $S = [S_t | S_d]$ (combination of both task and data parties' feature matrices), we want to leverage SVD to learn the latent representations,

$$S = U \Sigma V^T \tag{1}$$

where U is thus the latent representations of shared samples. Inspired by FedSVD [Chai *et al.*, 2021], we use a randomized masking method to learn U as,

1. A trusted key generator generates two randomized orthogonal matrices $A \in \mathbb{R}^{|I_s| \times |I_s|}$ and $B \in \mathbb{R}^{|X_{td}| \times |X_{td}|}$ ($|X_{td}| = |X_t| + |X_d|$). B is further partitioned to two parts $B_t \in \mathbb{R}^{|X_t| \times |X_{td}|}$ and $B_d \in \mathbb{R}^{|X_d| \times |X_{td}|}$, i.e., $B^T = [B_t^T | B_d^T]$.
2. A and B_t are sent to the task party; A and B_d are sent to the data party.
3. Each party does a local computation by masking their own feature matrices with the received matrices:
$$\hat{S}_k = A S_k B_k, \forall k \in \{t, d\} \tag{2}$$
4. Both task and data parties send $\hat{S}_t$ and $\hat{S}_d$ to a third-party server[3] and the third-party server runs SVD on the combined data matrix $\hat{S} = \hat{U} \Sigma \hat{V}^T$, where $\hat{S} = [\hat{S}_t | \hat{S}_d]$. $\hat{U}$ is then sent to the task party.
5. The task party can recover the federated latent representation of shared users, denoted as $\mathbf{x}_s^{fed}$, by
$$\mathbf{x}_s^{fed} = U = A^T \hat{U} \tag{3}$$

Compared to the original FedSVD which aims to recover both U and V [Chai *et al.*, 2021], we only need to recover U. Hence, in our method, only U' is transmitted to the task party to reduce the communication cost. The correctness of the above process depends on the fact that S and $\hat{S}$ (multiplying S by two orthogonal matrices) must hold the same singular value Σ. For the proof on the correctness and security of FedSVD, please refer to [Chai *et al.*, 2021].

[3]The third-party server needs to be semi-honest. Note that in FL, such a security configuration (i.e., the information aggregation server is semi-honests) is widely accepted [Yang *et al.*, 2019].

3.3 Local Representation Distillation

After obtaining $\mathbf{x}_s^{fed}$ for shared samples, Step 2 aims to enrich the task party's local sample representations. We thus design a novel local feature extracting strategy, which is combined with knowledge distilling from shared samples' $\mathbf{x}_s^{fed}$. Specifically, for a certain unsupervised local representation learner, we enhance it by adding a new loss function, i.e., making the shared samples I_s's learned representations be close to $\mathbf{x}_s^{fed}$, thus enabling the knowledge distillation effect.

In our mechanism implementation, we consider one of the most widely-used unsupervised representation extraction methods, *auto-encoder* [Hinton and Salakhutdinov, 2006]. Especially, if the input features are from a shared sample, we add a new knowledge distillation loss function by comparing the encoder's output to the shared sample's federated representation (learned from Step 1),

$$\mathcal{L}_{distill}(\mathbf{x}_s^t) = |Enc(\mathbf{x}_s^t) - \mathbf{x}_s^{fed}| \tag{4}$$

where $\mathbf{x}_s^t$ is the shared samples' local features in the task party. Hence, the complete loss function of the distilled auto-encoder is,

$$loss = \begin{cases} \mathcal{L}_{recons}(\mathbf{x}_i^t) + \theta \mathcal{L}_{distill}(\mathbf{x}_i^t) & i \in I_s \\ \mathcal{L}_{recons}(\mathbf{x}_i^t) & i \in I_t \setminus I_s \end{cases} \tag{5}$$

That is, for the task party's (non-shared) local samples, the loss function is the same as the original auto-encoder. For the shared samples, a new knowledge distillation loss is added to the original reconstruction loss; θ is the weight parameter to balance two loss function parts.

After training the federated-representation-distilled auto-encoder until convergence, the encoder part Enc can be a feature enrichment function for the task party's local samples. That is, for $i \in I_t \setminus I_s$, $Enc(\mathbf{x}_i^t)$ can be used to enrich the original local feature $\mathbf{x}_i^t$. In other words, the enriched local samples' representations $\mathbf{x}_i^* = \langle \mathbf{x}_i^t, Enc(\mathbf{x}_i^t) \rangle$ are given to Step 3 for training a task-specific machine learning model.

When there are n data parties, the task party can repeat the aforementioned Step 1 and 2 with each data party. Specifically, for each data party d_i, the task party can learn a local feature enrichment function Enc_i. Then, by aggregating n local feature enrichment functions learned from n data parties, the local samples' final enriched representations become,

$$\mathbf{x}_i^* = \langle \mathbf{x}_i^t, Enc_1(\mathbf{x}_i^t), Enc_2(\mathbf{x}_i^t), ..., Enc_n(\mathbf{x}_i^t) \rangle \tag{6}$$

3.4 Mechanism Updating

In general, our mechanism is efficient to update without the need of completely re-running three steps for all the parties.

- ***Local Incremental Learning*** *- New task party samples.* Note that the representation distillation is conducted locally at the task party. Then, if the task party t has a number of new local samples, t can locally re-conduct the representation distillation to learn an updated local

Dataset	#Samples	#Features	#Shared Samples
HAPT	5,000	250	3,000
Gisette	4,000	2,500	1,000
RNA-Seq	600	8,000	500
PCam16k	8,000	20,000	5,000
SUSY	1,500,000	6	20,000

Table 1: Default data samples and features held by a data party or the task party. The shared samples are the samples shared by the task party and an arbitrary data party.

feature enrichment function. The task party t does not need to communicate with any data parties for this updating, which is very efficient and convenient.[4]

- ***Task Independence*** - *New tasks*. Similar to new samples, if the task party t has a new task label to predict, t also does not need to communicate with other parties. t only needs to repeat Step 3 with the new task label.
- ***Knowledge Extensibility*** - *New data parties*. As we have discussed for the scenario of multiple data parties, it is easy to update our mechanism for new-coming data parties. In particular, the task party can learn a new local feature enrichment function Enc' from the new data party (repeat Step 1 and 2 with the new data party), and then enrich the local feature representation as $\mathbf{x}_i^* = \langle \mathbf{x}_i^*, Enc'(\mathbf{x}_i^t) \rangle$.

4 Evaluation

In this section, we empirically verify the effectiveness of our mechanism with three real-life datasets. Our experiments were performed on the workstation using Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, 50GB RAM, PyTorch 1.10.0, Python 3.8 and Cuda 11.3.

4.1 Datasets

We evaluate our mechanism on the following datasets.

- *Human Activities and Postural Transitions (HAPT)* [Reyes-Ortiz *et al.*, 2016] is an activity recognition dataset based on smartphone sensor readings. The dataset dimension is $\mathbb{R}^{10929\times561}$. The task label is the activity type (12 types).
- *Gisette* [Guyon *et al.*, 2006] is a handwritten digit dataset containing the confused '4' and '9' samples with 5000 features (pixels). The dimension is $\mathbb{R}^{13500\times5000}$. The task is a binary classification to identify '4' or '9'.
- *Gene Expression Cancer RNA-Seq (RNA-Seq)* [Chang *et al.*, 2013] dataset includes gene expressions in patients with different types of tumor. The dimension is $\mathbb{R}^{801\times20531}$ and there are 5 tumor types to predict.

[4]In practice, for new samples, the existing local feature enrichment function (without updating) can still be used. Our evaluation would test this setting (Sec. 4.5).

- *PCam16k* [Singh, 2021] is a supervised binary classification dataset published on OpenML with dimension $\mathbb{R}^{16000\times27648}$.
- *SUSY* [Baldi *et al.*, 2014] dataset describes a binary classification problem for distinguishing signal processes that produce supersymmetric particles from background processes that do not. Its dimension is $\mathbb{R}^{5000000\times19}$, where the first 8 features describe the kinematics of the particle and the last 10 features are functions of the first 8 features.

Table 1 shows the default data split to different parties in the experiments. We suppose that there exist one task party and one data party by default. Due to the page limitation, for most experiments, we show the results on HAPT and Gisette datasets.

4.2 Baselines

As far as we know, no existing methods are designed specifically for the multi-party vertical knowledge transfer problem (Sec. 2.2). Hence, to verify the effectiveness of our mechanism, we compare it with the baseline method using only the task party's local features. Moreover, we assume that samples' full features are known and thus build a baseline as the upper bound of the knowledge transfer performance.

- *LOCAL*: This baseline leverages only the task party's local features for training the task-specific model.
- *FULL*: For this baseline, we assume that *full* features of the task party's samples are known for training a machine learning model. This baseline is not a valid solution to our vertical federated knowledge transfer problem, but it can be seen as the upper bound of the knowledge transfer performance.

Note that we can leverage various machine learning algorithms to train the task-specific model. In our experiments, we use the random forest as the default machine learning algorithm. We also test the other popular algorithms including KNN, XGBoost [Chen and Guestrin, 2016], and neural networks for robustness check.

4.3 Training Configurations

We choose Adam optimizer for training auto-encoder, with learning rate = 0.001, batch size = 100, epoch = 20. The default layer number is 3 and the activation function is SIGMOID. Meanwhile, we also carry out experiments under the different numbers of hidden layers with three activation functions to verify the mechanism's robustness.

For all the datasets, when training the task-specific prediction model, we choose 80% of the data as the training set and 20% as the test set. In order to prevent the interference of random seeds, we carry out experiments under 30 different random seeds and compute the average results.

4.4 Main evaluation

We first report the results when there is only one data party. Fig. 2 and 3 show the prediction performance on HAPT by varying the number of features in the task and data party, respectively. Our mechanism can obtain the similar good performance on Gisette (Fig. 5 and 6). Results show that our

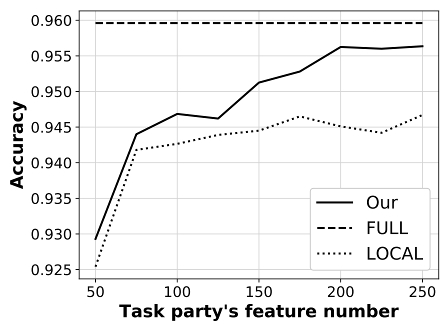

Figure 2: Prediction accuracy by varying the task party's feature number (HAPT).

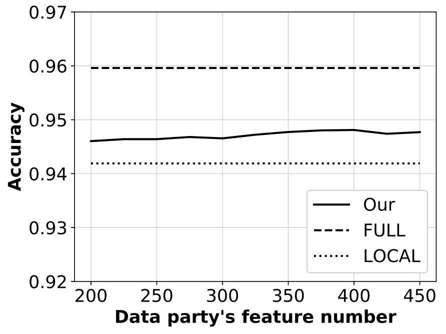

Figure 3: Prediction accuracy by varying the data party's feature number (HAPT).

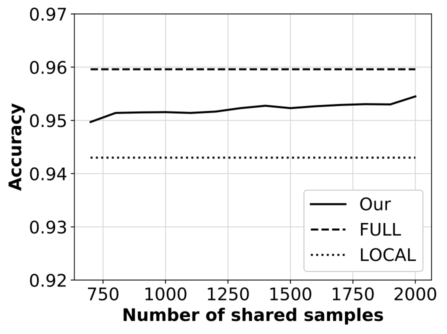

Figure 4: Prediction accuracy by varying the number of shared samples (HAPT).

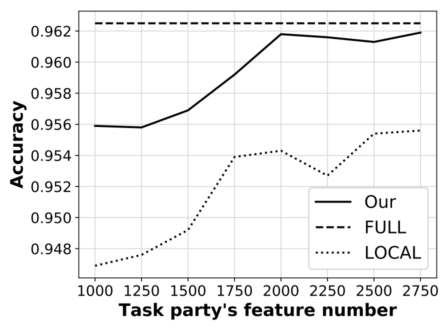

Figure 5: Prediction accuracy by varying the task party's feature number (Gisette).

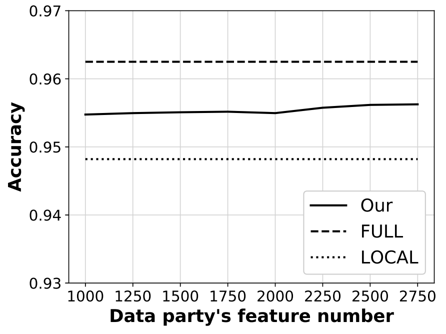

Figure 6: Prediction accuracy by varying the data party's feature number (Gisette).

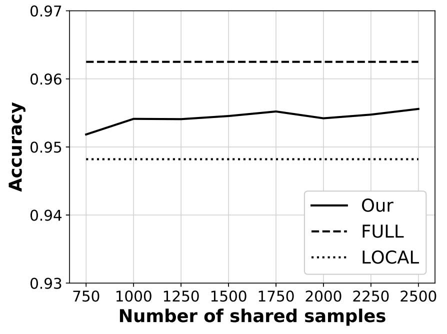

Figure 7: Prediction accuracy by varying the number of shared samples (Gisette).

#Method	RNA-Seq	PCam16k	SUSY
Our	0.9667	0.7425	0.7311
FULL	0.9689	0.7432	0.7990
LOCAL	0.9556	0.7250	0.7189

Table 2: Prediction accuracy on other datasets.

mechanism can consistently outperform the LOCAL baseline, which verifies the generalizable effectiveness of our vertical knowledge transfer mechanism. Besides, when the feature number increases (either the task party or the data party), our mechanism's accuracy gradually goes up.

Fig. 4 and 7 show how our mechanism performs by changing the number of shared samples between the task party and the data party. As expected, if there are fewer shared samples, the transfer performance degrades. This is because the knowledge transfer source is the shared samples — the more shared samples exist, the better knowledge transfer performs.

Fig. 8 and 9 explore our mechanism's performance with the different number of data parties. With the increase of data parties, the performance of our mechanism grows obviously. This means that our mechanism can obtain effective information from multiple data parties to make up for the shortcomings of insufficient data volume or fewer features.

In addition to HAPT and Giesette, Table 2 shows the experiment results on the other three datasets. Our mechanism consistently outperforms LOCAL, verifying the generalized effectiveness of our knowledge transfer method in various datasets.

4.5 Inductive Learning Results

Moreover, we check how our mechanism can facilitate inductive learning, i.e., new samples of the task party (the samples not used in training the encoder model). In reality, new samples are often with a different feature/label distribution from old samples since many factors may change with time going. We thus also purposely choose new samples so that their label distribution is obviously different from old samples as a non-IID experiment setting. Fig. 10 demonstrates that our mechanism can achieve better performance than LOCAL for both IID and non-IID new samples. Specifically, while the prediction accuracy decreases for both our method and LOCAL when the experiment setting changes from IID to non-IID, the loss of accuracy is much smaller for our method. This reveals the good generalizability of our mechanism's enriched representations.

4.6 Robustness Check

There are certain experimental hyper parameters. Here, we change such parameters to verify the robustness of our experimental results. Table 3 illustrates the prediction accuracy change when we modify the structure of auto-encoders in our mechanism. We can see that the resultant accuracy is robust to such modifications. Fig. 11 plots the prediction accuracy when applying different machine learning algorithms to train the task-specific prediction model. The results verify that our

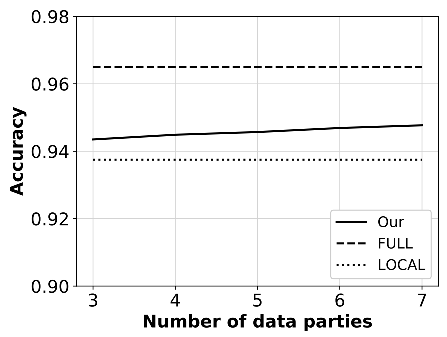

Figure 8: Prediction accuracy by varying the number of data parties (Gisette).

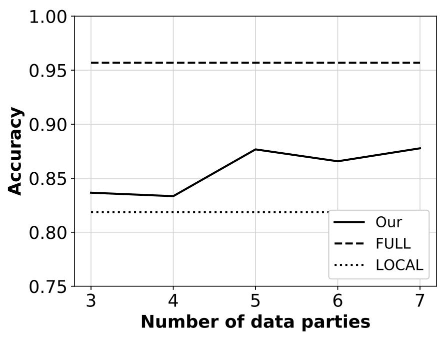

Figure 9: Prediction accuracy by varying the number of data parties (HAPT).

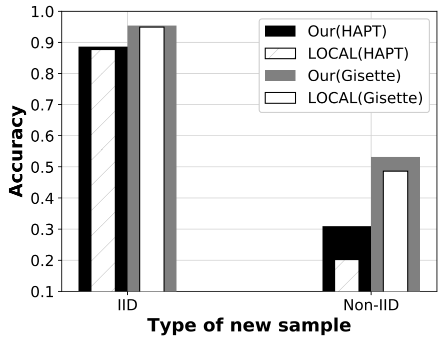

Figure 10: Prediction accuracy of new samples.

#Layers	Sigmoid	ReLU	Tanh	Leaky-ReLU
2	0.9504	0.9506	0.9501	0.9503
4	0.9510	0.9520	0.9504	0.9505
6	0.9501	0.9508	0.9505	0.9510
8	0.9503	0.9511	0.9509	0.9513
10	0.9505	0.9513	0.9513	0.9510
12	0.9516	0.9518	0.9510	0.9514

Table 3: Prediction accuracy by changing the layers and activation functions in the auto-encoder (Gisette).

mechanism can consistently outperform LOCAL under an arbitrary machine learning model, indicating the robustness of our knowledge transfer mechanism.

4.7 Viability Analysis

In order to further verify the effectiveness of our mechanism on knowledge transfer, we compared the changes in prediction accuracy before and after using our newly designed loss function, i.e., $|Enc(\mathbf{x}_s^t) - \mathbf{x}_s^{fed}|$. As Fig.12 shows, the prediction accuracy was significantly improved with the use of the novel loss component for knowledge transfer. Similarly, Fig.13 and 14 show that when the features of the task party change, our mechanism always performs better than the mechanism without knowledge transfer. The results show that the proposed penalty is effective and can carry out reasonable knowledge transfer. In particular, adding $|Enc(\mathbf{x}_s^t) - \mathbf{x}_s^{fed}|$ into the loss function of auto-encoder as penalty can provide a more reasonable auxiliary conversion means for knowledge transfer. $\mathbf{x}_s^{fed}$ can be regarded as the teacher and $Enc(\mathbf{x}_s^t)$ as the student. Students use their own data and the teacher's shared data $\mathbf{x}_s^{fed}$ to conduct knowledge distillation. This process makes the generated representations benefit from the teacher's shared knowledge (which is learned from both task party and data party's raw features).

4.8 Computation Time

We record the computation time of our mechanism by varying the number of data parties and the number of shared samples. To keep the computation time to an order of magnitude, we calculated the time to run our mechanism 10 times on HAPT dataset. The results are shown in Fig. 15 and 16, respectively. In general, the computation time of our mechanism is linearly proportional to the number of data parties and the number of shared samples. This linear relationship indicates the good scalability of our mechanism.

5 Related Work

Vertical FL focuses on cross-organization collaborative learning. The common setting is that different organizations hold different features of the same set of samples, which is also often known as vertical FL [Yang *et al.*, 2019]. Another mainstream of FL algorithms, known as horizontal FL, have general-purpose optimization algorithms such as FedAvg [McMahan *et al.*, 2017] for building various machine learning models; however, for vertical FL, up to date, there is no such general-purpose learning algorithms. In particular, for different machine learning models, researchers have proposed diverse mechanisms, such as tree-based models [Cheng *et al.*, 2019; Wu *et al.*, 2020] and neural networks [Hu *et al.*, 2019].

Compared to these existing vertical FL algorithms, the key difference of our mechanism is the application scope. Existing vertical FL algorithms focus on improving the prediction performance on shared samples. In contrast, our mechanism aims to improve the prediction performance of each party's local (non-shared) samples by transferring the knowledge from shared samples. We believe that our mechanism can be a good complement to existing vertical FL algorithms, thus boosting the practicability of FL in reality.

A prior study close to our research is the FTL (federated transfer learning) framework [Liu *et al.*, 2020]. However, our knowledge transfer process is *task-independent*, which means that the distilled representation of the task party's samples (i.e., learned from the distilled encoder) can benefit an arbitrary machine learning task for the task party; in comparison, FTL works only for a predefined machine learning task.

6 Conclusion

In this paper, we propose a vertical federated knowledge transfer mechanism to transfer the knowledge from cross-party shared samples to each party's local samples. Our

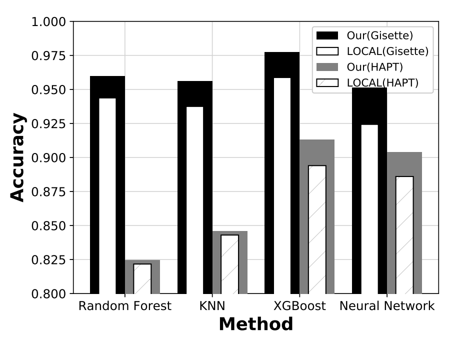

Figure 11: Prediction accuracy of different machine learning models.

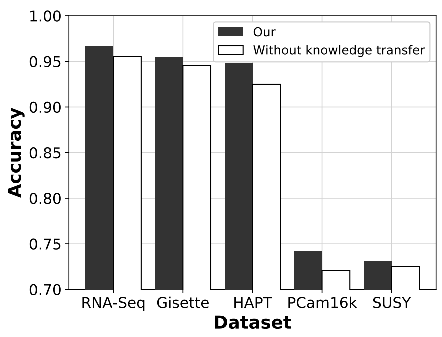

Figure 12: Changes in prediction accuracy with different datasets before and after knowledge transfer.

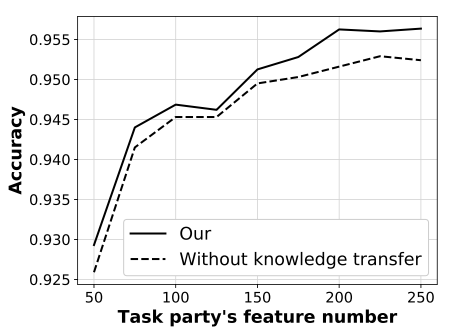

Figure 13: Changes in prediction accuracy with different task party's feature number before and after knowledge transfer (HAPT).

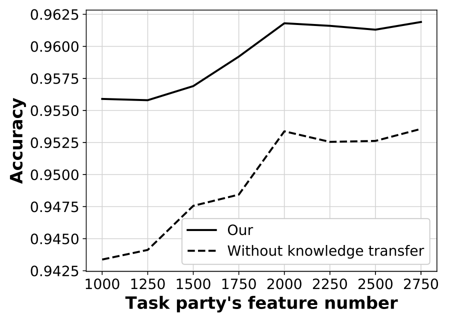

Figure 14: Changes in prediction accuracy with different task party's feature number before and after knowledge transfer (Gisette).

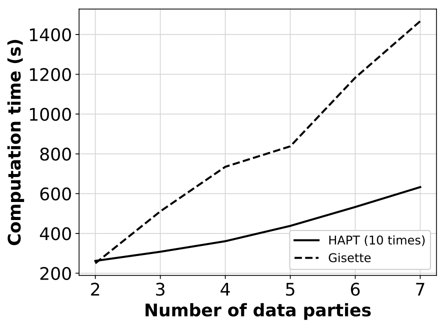

Figure 15: Computation time by varying the number of data parties.

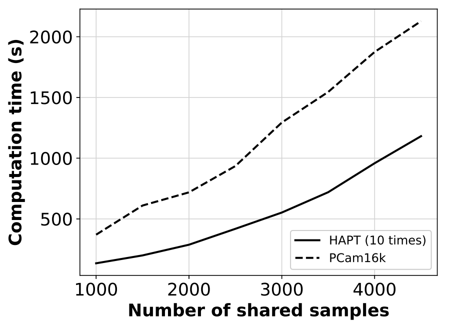

Figure 16: Computation time by varying the number of samples per party.

mechanism can significantly improve the application scenarios of vertical FL as it is complementary to the traditional solutions that only work for shared samples. Experiments on five real-life datasets and varying configurations verify the effectiveness of our mechanism.

In the future, we aim to explore more possible techniques used in Step 1 and 2 of our mechanism to improve the knowledge transfer performance. For instance, contrastive learning has become a popular way for self-supervised representation learning, and researchers have recently studied certain federated contrastive learning methods [Li *et al.*, 2021]. We would like to explore whether it is possible to incorporate federated contrastive learning into our mechanism for further enriching local samples' representations.

References

[Baldi *et al.*, 2014] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

[Chai *et al.*, 2021] Di Chai, Leye Wang, Lianzhi Fu, Junxue Zhang, Kai Chen, and Qiang Yang. Federated singular vector decomposition. *ArXiv*, abs/2105.08925, 2021.

[Chang *et al.*, 2013] Kyle Chang, Chad J Creighton, Caleb Davis, Lawrence Donehower, Jennifer Drummond, David Wheeler, Adrian Ally, Miruna Balasundaram, Inanc Birol, Yaron SN Butterfield, et al. The cancer genome atlas pan-cancer analysis project. *Nat Genet*, 45(10):1113–1120, 2013.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[Cheng *et al.*, 2019] K. Cheng, T. Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *ArXiv*, abs/1901.08755, 2019.

[Guyon *et al.*, 2006] Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Feature selection with the clop package. Technical report, Technical report, 2006.

[Hinton and Salakhutdinov, 2006] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504 – 507, 2006.

[Hinton *et al.*, 2015] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[Hu *et al.*, 2019] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2232–2240, 2019.

[Kamara *et al.*, 2014] S. Kamara, Payman Mohassel, Mariana Raykova, and Seyed Saeed Sadeghian. Scaling private set intersection to billion-element sets. In *Financial Cryptography*, 2014.

[Kosinski *et al.*, 2013] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the national academy of sciences*, 110(15):5802–5805, 2013.

[Li *et al.*, 2021] Qinbin Li, Bingsheng He, and Dawn Xiaodong Song. Model-contrastive federated learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10708–10717, 2021.

[Liu *et al.*, 2020] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35:70–82, 2020.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Reyes-Ortiz *et al.*, 2016] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[Singh, 2021] Prabhant Singh. Pcam16k. https://www.openml.org/search?type=data&sort=runs&id=42811&status=active, 2021. Accessed: 2021-03-08.

[Wu *et al.*, 2020] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *VLDB*, 13(11):2090–2103, 2020.

[Yang *et al.*, 2018] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12, 2019.