FedMVT: Semi-supervised Vertical Federated Learning with MultiView Training

Yan Kang, Yang Liu, Tianjian Chen

1 WeBank, Shenzhen, China
{yangkang, yangliu, tobychen}@webank.com

Abstract

Federated learning allows many parties to collaboratively build a model without exposing data. Particularly, vertical federated learning (VFL) enables parties to build a robust shared machine learning model based upon distributed features about the same samples. However, VFL requires all parties to share a sufficient amount of overlapping samples. In reality, the set of overlapping samples may be small, leaving the majority of the non-overlapping data unutilized. In this paper, we propose Federated Multi-View Training (FedMVT), a semi-supervised learning approach that improves the performance of VFL with limited overlapping samples. FedMVT estimates representations for missing features and predicts pseudo-labels for unlabeled samples to expand training set, and trains three classifiers jointly based upon different views of the input to improve model’s representation learning. FedMVT does not require parties to share their original data and model parameters, thus preserving data privacy. We conduct experiments on the NUS-WIDE and the CIFAR10. The experimental results demonstrate that FedMVT significantly outperforms vanilla VFL that only utilizes overlapping samples, and improves the performance of the local model in the party that owns labels.

1 Introduction

With the increasingly stricter privacy-protection laws implemented worldwide [GDPR, 2018; DLA Piper, 2018], federated learning has received significant attention and became a popular research topic recently [Kairouz et al., 2019]. With the research goes deeper and wider, the practice of federated learning has been expanded from building powerful mobile applications based on data resided in millions of mobile devices [McMahan et al., 2016] to solving the problem of data fragmentation and isolation among or within organizations [Yang et al., 2019a]. For example, many business decisions of a bank may rely on the purchasing preferences of its customers. This bank may share some customers with a local retail company, who owns purchasing preference data of the local people. Thus, the bank can invite the retail company to collaboratively build a shared model leveraging the data features of both sides to improve its businesses. The retail company can also benefit from the shared model.

This and many other similar practical demands [Yang et al., 2019a] motivate the development of vertical federated learning (VFL) [Yang et al., 2019b] that enables participating parties to collaboratively train a machine learning model by fully utilizing scattered features of their overlapping samples without exposing data. However, a critical prerequisite of VFL is that it requires all parties to share a sufficient amount of overlapping samples in order to achieve decent performance. [Liu et al., 2018] propose a federated transfer learning framework to address weak supervision (few labels) problems in the VFL setting. Nonetheless, it does not utilize all labeled data for supervised training, nor does it take full advantage of unlabeled samples for improving learning quality. Many other similar approaches such as domain adaptation [Peterson et al., 2019] and knowledge distillation [Li and Wang, 2019], have been applied in the federated learning setting. However, they mainly focus on scenarios where all parties share the same feature space (also known as horizontal federated learning [Yang et al., 2019b]). Therefore, research on applying semi-supervised techniques in the VFL setting is insufficient.

In this paper, we propose a novel semi-supervised algorithm in VFL setting, termed FedMVT, to address the limitations of existing vertical federated learning approaches. Our contributions are as follows:

1. FedMVT significantly boosts up the performance of the federated model when the overlapping samples between parties are limited;
2. FedMVT helps improve the performance of the local classifier of the party that owns labels, and thus it enables the party to make local predictions with more confidence;
3. FedMVT works with data of various types. This is important in practice since real-world VFL applications often need to deal with heterogeneous features;
4. FedMVT does not require participating parties to share their original data and model parameters, but only intermediate representations and gradients, which can be further protected by the VFL-DNN (Deep Neural Network) framework implemented by FATE.

2 Related Work

In this section, we focus on reviewing closely related fields and approaches.

Vertical Federated Learning (VFL). VFL is also known as feature-partitioned machine learning in some literature. The secure linear machine learning with data partitioned in the feature space has been well studied in [Gascón et al., 2016; Hardy et al., 2017; Mohassel and Zhang, 2017]. They apply either hybrid MPC (Secure Multi-party Computation) protocol or additively homomorphic encryption [Rivest et al., 1978] for secure linear model training. [Cheng et al., 2019] proposed a secure federated tree-boosting (SecureBoost) approach in the VFL setting that enables participating parties with different features to collaboratively build a set of boosting trees, and they prove that the SecureBoost provides the same level of accuracy as its non-privacy-preserving centralized counterparts. FATE designed and implemented a VFL-DNN (Deep Neural Network) framework that supports DNN in the VFL setting. VFL-DNN leverages a hybrid encryption scheme in the forward stage and backward stage of the VFL training process to protect data privacy and model security. [Vepakomma et al., 2018] proposed several configurations of splitting a deep neural network to support various forms of collaborations among health entities that each holds a partial deep network and owns a different modality of patient data.

Semi-supervised Learning (SSL). SSL aims to alleviate the need for a large amount of labeled data by allowing a model to leverage unlabeled data. Many recent approaches for semi-supervised learning use transfer learning [Pan et al., 2010], consistency regularization [Tarvainen and Valpola, 2017; Laine and Aila, 2017] or pseudo-labeling [Hyn Lee, 2013; Clark et al., 2018] to learn from unlabeled data. [Berthelot et al., 2019] proposed MixMatch that unifies three dominant semi-supervised methods: entropy minimization [Grandalet and Bengio, 2004], consistency regularization and generic regularization. They demonstrate that MixMatch can approach the error rate achieved by fully supervised training with significantly fewer data. [Clark et al., 2018] proposed a semi-supervised Cross-View Training (CVT) approach that simultaneously trains a full model based upon all labeled data and multiple auxiliary models that only see restricted views of the unlabeled data. During training, CVT teaches auxiliary models to match predictions made by the full model to improve representation learning and in turn improve the full model. While successful, these works are not designed for VFL scenarios where features are scattered among parties. [Liu et al., 2018] proposed a secure Federated Transfer Learning (FTL) framework, which is the first framework that enables VFL to benefit from transfer learning. FTL helps the party in the target domain build a prediction model by utilizing rich label resources from the source domain.

3 Problem Definition

Consider party A having dataset $D^A := \{ (x^{A,i}, y^{A,i}) \}_{i=1}^{N^A}$ where $x^{A,i} \in R^n$ is the feature of the i-th sample and $y^{A,i} \in \{0, 1\}^C$ is the corresponding one-hot encoding ground-truth label for C classes corresponding to $x^{A,i}$. Party B having $D^B := \{ x^{B,i} \}_{i=1}^{N^B}$ where $x^{B,i} \in R^n$. $D^A$ and $D^B$ are held privately by the two parties and cannot be exposed to each other. $N^A$ and $N^B$ are numbers of samples for $D^A$ and $D^B$ respectively.

We assume that there exists a limited set of overlapping samples $D_{ol} := \{ x^{ol,i} \}_{i=1}^{N_{ol}}$ between the two parties, where party B owns the partition $D_{ol}^B := \{ x^{B,i} \}_{i=1}^{N_{ol}}$ and party A owns the rest $D_{ol}^A := \{ x^{A,i} \}_{i=1}^{N_{ol}}$. $N_{ol}$ is the number of overlapping samples. One can find the set of overlapping samples through encrypted entity alignment techniques in a privacy-preserving setting [Nock et al., 2018]. Here, we assume that party A and party B have already found or known the IDs of their overlapping samples. We denote $D_{nl}^A := \{ x^{A,i} \}_{i=1}^{N_{nl}}$ as the non-overlapping samples for party A, while $D_{nl}^B := \{ x^{B,i} \}_{i=1}^{N_{nl}}$ for party B.

If we concatenate $D^A$ and $D^B$ together as one big virtual dataset in a tabular view, this virtual dataset is vertically partitioned, and each party owns one vertical partition of this dataset. This is where the term "vertical federated learning" comes from. Figure 1 shows the tabular view of the vertically partitioned virtual dataset owned by two parties, A and B.

![Figure 1: View of the virtual dataset in Vertical Federated Learning. Each party owns a vertical partition of this dataset.](https://github.com/FederatedAI/FATE)

The vanilla VFL is trying to build a federated machine learning model utilizing only overlapping samples $D_{ol}$, leaving non-overlapping samples $D_{nl}^A$ and $D_{nl}^B$ unused. We, therefore, propose a semi-supervised VFL with Multi-View Training (FedMVT) approach that not only fully utilizes all available data but also estimates representation of missing features (shown in figure 1) for not only significantly improving the performance of the federated model but also helping improve the performance of the local model of party A.

4 The Proposed Approach

Deep neural networks have been widely used to learn feature representation [Oquab et al., 2014]. For each party, we uti-
lize two neural networks to learn feature representations from raw input data. One is to learn feature representations weakly shared between the two parties and the other is to learn feature representations unique to each party.

More specifically, we denote $r^p_u = h^p_u(x^p)$ as the unique feature representations and $r^p_r = h^p_r(x^p)$ as the shared feature representations that are learned from features $x^p$ through neural networks $h^p_u$ and $h^p_r$ respectively in party $p \in \{A, B\}$, where $r^p_u \in \mathbb{R}^{N^p \times d^p}$ and $r^p_r \in \mathbb{R}^{N^p \times d^p}$, and $d^p$ is the dimension of the top hidden representation layer of neural networks in party $p$. Then, the complete feature representations for $x^p$ is denoted as $x^p = [r^p_u; r^p_r]$, where $[;]$ is the concatenation operator that concatenates two matrices along the feature axis.

For convenience, we denote $r^p_{u, ol}$ and $r^p_{c, ol}$ as the feature representations learned from overlapping samples $x^p_{ol}$ in party $p \in \{A, B\}$ (Figure 2(a)), while $r^p_{u, nl}$ and $r^p_{c, nl}$ as the feature representations learned from non-overlapping samples $x^p_{nl}$ (Figure 2(b) and 2(c)). Correspondingly, the complete feature representations for $x^p_{ol}$ and $x^p_{nl}$ are denoted as $r^p_{ol} = [r^p_{u, ol}; r^p_{c, ol}]$ and $r^p_{nl} = [r^p_{u, nl}; r^p_{c, nl}]$, respectively.

Intuitively, $h^p_r$ aims to capture common feature representations between the two parties while $h^p_u$ helps learn domain-specific representations. We propose the following three loss terms to enforce neural networks to learn the desired representations.

\[
L^{AB}_{dist}(r^A_{c, ol}; r^B_{c, ol}) = \frac{1}{N_{ol}} \sum_i \left\| r^{A,i}_{c, ol} - r^{B,i}_{c, ol} \right\|_F^2
\]  

\[
L^A_{orth}(r^A_u; r^A_c) = \frac{1}{N^A} \sum_i \left\| r^{A,i}_u \otimes r^{A,i}_c \right\|_F^2
\]

\[
L^B_{orth}(r^B_u; r^B_c) = \frac{1}{N^B} \sum_i \left\| r^{B,i}_u \otimes r^{B,i}_c \right\|_F^2
\]

where $\otimes$ denotes matrix multiplication operator and $\| \cdot \|_F^2$ is the squared Frobenius norm. By minimizing (1), neural networks $h^A_u$ and $h^B_u$ are pushed to learn common feature representations from the raw data of the two parties. While (2) and (3) are orthogonality constraints [Bousmalis et al., 2016] that encourage neural networks $h^A_r$ and $h^B_r$ belonging to party $p \in \{A, B\}$ to learn distinct feature representations.

Inspired by multi-view learning [Clark et al., 2018] to share representations across models and improve models’ representation learning, we train three softmax classifiers $f^A$, $f^B$ and $f^{AB}$ jointly and each takes as input a feature representation learned from a different view of the virtual dataset (Figure 1) and outputs estimated class distributions. The loss functions for the three classifiers are defined as follows:

\[
L^A(r^A, y^A) = \frac{1}{N^A} \sum_i L_{ce}(f^A(r^{A,i}), y^{A,i})
\]

\[
L^B(r^B, y^A) = \frac{1}{N^B} \sum_i L_{ce}(f^B(r^{B,i}), y^{A,i})
\]

\[
L^{red}(r^B, y^A) = \frac{1}{N_{ol}} \sum_i L_{ce}(f^{AB}(r^{B,i}_r; r^{A,i}_c), y^{A,i})
\]
Formulas (4), (5) and (6) compute the cross-entropy between estimated class distributions and ground-truth labels. Then, we have the objective loss:

\[ L_{\text{obj}} = L^\text{fed} + L^A + L^B + \lambda_1 L^\text{dist} + \lambda_2 L^\text{orth} + \lambda_3 L^\text{orth} \]  

where \( \lambda_1 \) are loss weights. Formula (7) forms the backbone of the FedMVT approach. For now, it only uses existing data without considering missing features and labels. In sections 4.1 and 4.2, we will elaborate on how we estimate representations of missing features and determine pseudo-labels for unlabeled samples. The workflow is overviewed in figure 2.

4.1 Feature Representation Estimation

We estimate representations for the missing features in the virtual dataset (figure 1) by adopting the scaled dot-product attention function [Vaswani et al., 2017]. We explain how the estimation is calculated by walking through the procedure of estimating representations for missing features in party A (figure 2b).

Formula (8) estimates the shared representation, denoted as \( r^A_{c,i} \), for party A’s missing feature corresponding to the \( i \)th sample \( x^{B,i} \) of party B by leveraging shared feature representation \( r^c_{B} \) learned from \( x^{B,i} \) and \( r^c_{A} \) learned from \( x^A \).

\[
\hat{r}^A_{c,i} = g^A_c(r^B_{c,i}) = softmax\left(\frac{r^{B,i}_c \otimes r^A_c}{\sqrt{d}} \right) \otimes r^c_{A} \tag{8}
\]

Intuitively, \( g^A_c \) can be interpreted as a search engine such that given \( r^B_{c,i} \) by party B, it gives a result of \( \hat{r}^A_{c,i} \) in party A.

Figure 3 illustrates the steps of applying formula (8) to estimate \( \hat{r}_{c,nl}^A \) for party A’s missing feature corresponding to the 3rd non-overlapping feature \( x_{n}^{B,3} \) of party B: first, original features \( x^{B,3} \) and \( x^{A,3} \) are transformed into corresponding shared representations \( r^{B,3}_c \) and \( r^{A,3}_c \) in a common feature representation space through neural networks \( h^B_{c} \) and \( h^A_{c} \). Then, the similarities between representation \( r^{B,3}_c \) and the other two (i.e., \( r^{A,3}_c \) and \( r^{c}_A \)) are calculated, denoted as \( s_1 \) and \( s_2 \) respectively. Further, step 3 is applied to calculate \( r_{c,nl}^{A,3} \).

We apply formula (9) to estimate unique representation, denoted as \( \tilde{r}_{u}^{A,i} \), for party A’s missing feature corresponding to the \( i \)th feature \( x^{B,i} \) of party B by leveraging unique feature representations \( r_{B,i}^{u} \) and \( r_{u,ol}^{B} \) learned from \( x^{B,i} \) and \( x_{ol}^{B} \) respectively in party B, and \( r_{u,ol}^{A} \) learned from \( x_{ol}^{A} \) in party A.

\[
r_{u}^{A,i} = g_{u,c}^{A}(r_{u,ol}^{B}) = softmax\left(\frac{r_{u,ol}^{B} \otimes r_{u,ol}^{A}}{\sqrt{d}} \right) \otimes r_{u,ol}^{A} \tag{9}
\]

Figure 4 illustrates the steps of applying formula (9) to estimate representation \( \hat{r}_{u,nl}^{A,3} \) for party A’s missing feature corresponding to feature \( x_{n}^{B,3} \). First, the original features from each party are projected into a unique feature representation space. Then, the similarities between representation \( r_{u,3} \) and the other two (i.e., \( r_{u,1}^{A,i} \) and \( r_{u,2}^{A,i} \)) in space B are calculated, denoted as \( s_1 \) and \( s_2 \) respectively. Further, \( s_1 \) and \( s_2 \) are transferred to space A. Specifically, \( s_1 \) is taken as similarity between \( r_{u,3} \) and \( r_{u,ol}^{A} \) while \( s_2 \) is taken as similarity between \( r_{u,3} \) and \( r_{u,ol}^{A} \). Last, step 4 is applied to calculate \( \hat{r}_{u,nl}^{A,3} \).

Following the same logic, we can estimate shared representation \( \hat{r}_{u,nl}^{B,3} = g_{u,c}^{B}(r_{u,3}^{B}) \) and unique representation \( \hat{r}_{u,nl}^{B,3} = g_{u,c}^{B}(r_{u,3}^{B}) \) for party B’s missing feature corresponding to the feature \( x_{b,3} \) of party A.

Thereby, we obtain representations \( r_{n}^{A} = \{\hat{r}_{u,nl}^{A,i} : \hat{r}_{c,nl}^{A}\} \) and \( r_{n}^{B} = \{\hat{r}_{u,nl}^{B,i} : \hat{r}_{c,nl}^{B}\} \) estimated for missing features of party A and party B, respectively, and construct the enlarged training set \( \chi \) consisting of three parts: \( \chi_{nl}^{A} = \{r_{u,1}^{A,i} \otimes y_{nl}^{A,i} \otimes x_{nl}^{A,i} \} \) and \( \chi_{nl}^{B} = \{r_{u,1}^{B,i} \otimes y_{nl}^{B,i} \otimes x_{nl}^{B,i} \} \), where \( \chi_{nl}^{A} \) and \( \chi_{nl}^{B} \) have ground-truth labels located in party A, while \( \chi_{nl}^{A} \) has no labels. We will estimate labels for \( \chi_{nl}^{A} \) in section 4.2.
We add two loss terms to force the estimated feature representations to approximate the representations learned from original features: \( L_{dist}^A(r^A_\alpha, r^A_\beta) = \frac{1}{N_{\alpha}} \sum_{i} \| r^A_{\alpha,i} - r^A_{\beta,i} \|_F^2 \), \( L_{dist}^B(r^B_\alpha, r^B_\beta) = \frac{1}{N_{\beta}} \sum_{i} \| r^B_{\alpha,i} - r^B_{\beta,i} \|_F^2 \), where \( r^p \in \{A, B\} \) is the representations estimated for overlapping samples in party \( p \) and it should be as close to \( r^p_\alpha \) as possible.

### 4.2 Pseudo-label Prediction

As discussed in section 4.1, \( \chi_{nl}^B = \{ x_{nl}^A; z_{nl}^A \} \) has no labels. We apply the three trained softmax classifiers \( f^A, f^B \) and \( f^{AB} \) that take as inputs \( x_{nl}^A, z_{nl}^A \) and \( x_{nl}^B, z_{nl}^A \) respectively to produce three candidate pseudo-labels for each unlabeled sample in \( \chi_{nl}^B \) (algorithm 1, line 6). Only when at least two of the three candidate pseudo-labels are equal and their probabilities are higher than a predefined threshold \( t \), we add this pseudo-labeled sample to the training set (algorithm 1, line 10). With all samples in \( \chi \) have labels, \( f^A, f^B \) and \( f^{AB} \) are trained on training sets \( \chi_{nl+nl}^A, \chi_{nl+nl}^B \) and \( \chi \) respectively, where \( \chi_{nl+nl}^A \) is the combination of \( \chi_{nl}^A \) and \( \chi_{nl+nl}^A \) and \( \chi_{nl+nl}^B \) is the combination of \( \chi_{nl}^B \) and \( \chi_{nl+nl}^B \).

Algorithm 1 gives an overview of the full FedMVT algorithm, in which the \( \text{concat} \) function concatenates input matrices along the feature axis and the \( \text{combine} \) function concatenates the input matrices along sample axis.

### 4.3 Security Analysis

FedMVT does not require parties to share their original data and model parameters, but only intermediate representations and gradients. The intermediate representations are results transformed from original features through deep neural networks with multiple layers of transformation functions. Therefore, there is little chance for the other party to reverse-engineer the original features [Gupta and Raskar, 2018]. Recently, there has been much discussion on the potential risks associated with privacy leakage through gradients [Bonawitz et al., 2017; Phong et al., 2018]. To prevent intermediate representations and gradients from being exposed, FATE designed and implemented a VFL-DNN (Deep Neural Network) framework that can efficiently perform encryption in both the forward stage and backward stage of the VFL training process. For experimental convenience, we did not implement FedMVT with FATE. However, FedMVT is compatible with the VFL-DNN framework and can be migrated to FATE with no much effort.

### 5 Experimental Evaluation

We report experiments conducted on public datasets including: 1) NUS-WIDE dataset [Chua et al., 2009] 2) CIFAR-10 [Krizhevsky, 2009] to validate our proposed approach.

**NUS-WIDE.** The NUS-WIDE dataset consists of 634 low-level image features extracted from Flickr images and their associated 1000 textual tags as well as 81 ground truth labels. Here, we consider solving a 10-label classification problem with a data federation formed between party A and party B. Each party utilizes two local neural networks that each has one hidden layer with 32 units to learn feature representations from their raw inputs. Then, each party feeds the learned feature representations into its local softmax classifier \( f^p, p \in \{A, B\} \) and the federated softmax layer \( f^{AB} \) with 128 (32x2x2) hidden nodes, respectively, for jointly training.

On NUS-WIDE, we run experiments with two scenarios. In the first scenario, denoted as scenario-1, we put 634 image features on party A and 1000 textual features on party B. The second scenario, denoted as scenario-2, is the other way around. In both scenarios, party A owns the labels.

**CIFAR-10.** We partition each CIFAR-10 image with shape \( 32 \times 32 \times 3 \) vertically into two parts (each part has shape \( 32 \times 16 \times 3 \)). Each party uses two local VGG-like CNN models to learn representations from raw image features, and then it feeds the learned representations into its local softmax classifier \( f^p, p \in \{A, B\} \) and the federated softmax classifier \( f^{AB} \), respectively, for jointly training. The VGG-like CNN model consists of 2x2 max-pooling layers, 3x3 convolutional layers with stride 1 and padding 1, and fully connected layers. The architecture is \( \text{conv32-conv32-pool-conv64-conv64-pool-conv128-conv128-pool-fc64} \). Thus, the federated softmax classifier has 256 (64x2x2) nodes.

**FedMVT and Baselines.** FedMVT-VFL and FedMVT-local are \( f^{AB} \) and \( f^A \), respectively, trained with algorithm 1. Vanilla-VFL is \( f^{AB} \) trained on overlapping samples, while

---

**Algorithm 1 FedMVT algorithm**

1. **Input:**
   - Datasets \( D^A \) and \( D^B \);
   - Batch index sets \( T_{ol}^A \) and \( T_{ol}^B \) of \( D^A \) and \( D^B \) respectively.
   - Neural networks \( h^B, h^A \) and \( h^C \);
   - Representation estimators \( g^B, g^A \) and \( g^A \);
   - Softmax classifiers \( f^A, f^B \) and \( f^{AB} \);

2. **Class probability threshold** \( t \) and the epoch number \( K \)

3. **Select a mini-batch:**
   - \( \{ x_{nl}^A, y_{nl}^A \}, i \in T_{ol}^A \); \( \{ x_{nl}^B, y_{nl}^B \}, j \in T_{ol}^B \); \( \{ x_{nl}^C, y_{nl}^C \} \)

4. **Learn** \( r_{nl}^A, r_{nl}^B, r_{nl}^C \) through \( h^B, h^A, h^C \);

5. **Estimate** \( r_{nl}^A, r_{nl}^B \), \( r_{nl}^C \) through \( g^B, g^A, g^C \);

6. **Estimate pseudo-labels:**
   - \( \tilde{y}_{nl}^A = f^A(\bar{r}_{nl}^A), \tilde{y}_{nl}^B = f^B(\bar{r}_{nl}^B), \tilde{y}_{nl}^C = f^{AB}(\bar{r}_{nl}^C) \);

7. **Select** \( \tilde{y}_{nl}^A, \tilde{y}_{nl}^B, \tilde{y}_{nl}^C \);

8. **Combine** \( \tilde{y}_{nl}^A, \tilde{y}_{nl}^B, \tilde{y}_{nl}^C \);

9. **Compute loss:**
   - Multiclass classification loss \( L_{obj} = L_{fled} + L + L + \lambda_1 L_{dist} + \lambda_2 L_{dist} + \lambda_2 L_{dist} + \lambda_3 L_{orth} + \lambda_3 L_{orth} \);

10. **Update gradients**
    - \( \nabla L_{obj} \) and update model parameters.

11. **end for**
Vanilla-local is \( f^A \) trained on local data of party A. For each dataset, they use the same local neural networks.

For NUS-WIDE, figure 5 shows the accuracy comparison of FedMVT to vanilla models with a varying number of overlapping samples from 250 to 8000. For both scenarios, FedMVT-VFL outperforms vanilla-VFL by a significantly large margin. For example, with 250 overlapping samples, FedMVT-VFL makes improvements of 34 and 39 points in accuracy over vanilla-VFL in scenario-1 and scenario-2, respectively. With 8000 overlapping samples, FedMVT-VFL improves the accuracy by 3.5 and 5.9 points compared with vanilla-VFL, respectively. FedMVT also enhances the performance of the vanilla-local model of party A. With 8000 overlapping samples, FedMVT-local outperforms vanilla-local by 3.2 and 4.6 points in scenario-1 and scenario-2 respectively, which demonstrates that the local model performs increasingly well with better feature representation.

As shown in table 1, scenario-2 outperforms scenario-1 across all models, which demonstrates that textual features have more discriminative power than image features on the NUS-WIDE. This partially explains why VFL models do not outperform their corresponding local models (blue v.s. red in figure 5(b)) in scenario-2 when the size of overlapping samples is not sufficiently large enough (e.g., \( \leq 2000 \)). While in scenario-1, party A benefits significantly from applying VFL (blue v.s. red in figure 5(a)). This suggests that, in practice, when the size of the overlapping samples is limited, the party owning labels may be better off using its local model if it has features with strong discriminative power. Otherwise, it may benefit from leveraging features of the other party.

Figure 5: Test accuracy (%) comparison of FedMVT models to vanilla models on NUS-WIDE for a varying number of overlapping samples. Exact numbers are provided in table 1. The number of local data is slightly different for different size of overlapping samples, and is around 30000 \( \pm \) 2000. (a) shows results when party A holds image data, while (b) shows results when party A hold text data.

![Figure 5](image_url)

Table 1: Test accuracy (%) comparison on NUS-WIDE corresponding to figure 5.

| Model         | 250   | 500   | 1000  | 2000  | 4000  | 8000  |
|---------------|-------|-------|-------|-------|-------|-------|
| Vanilla-Local | 58.3  | 58.3  | 58.5  | 58.8  | 58.9  | 58.9  |
| Vanilla-VFL   | 29.7  | 49.0  | 63.9  | 68.3  | 70.9  | 71.5  |
| FedMVT-local  | 57.2  | 58.7  | 59.0  | 59.6  | 60.1  | 62.1  |
| FedMVT-VFL    | 63.8  | 67.3  | 68.4  | 71.7  | 74.4  | 75.0  |

For CIFAR10, FedMVT-powered models also outperform vanilla models. For example, with 8000 overlapping samples (table 2), the FedMVT-VFL and FedMVT-local improve the accuracy by 8.5 and 1.9 points compared with vanilla-VFL and vanilla-local, respectively.

![Figure 6](image_url)

Figure 6 shows that the vanilla-VFL cannot beat vanilla-local with 10000 overlapping samples (over 1/3 of total samples). One explanation for this is that VFL only leverages the compressed information of the top dense layer of local CNN models of the two parties, thereby losing many useful feature patterns, for trading minimal information leakage. FedMVT equips VFL with an effective way of improving models’ representation learning.

Table 2: Test accuracy (%) comparison on CIFAR10 corresponding to figure 6.

| Model         | 500   | 1000  | 2000  | 4000  | 6000  | 8000  |
|---------------|-------|-------|-------|-------|-------|-------|
| Vanilla-local | 63.4  | 64.4  | 65.3  | 65.8  | 66.2  | 67.2  |
| Vanilla-VFL   | 40.0  | 44.2  | 50.4  | 56.5  | 60.7  | 62.9  |
| FedMVT-local  | 61.3  | 64.0  | 64.9  | 66.9  | 68.0  | 69.1  |
| FedMVT-VFL    | 62.7  | 65.0  | 66.0  | 67.8  | 70.5  | 71.4  |

The experimental setup on CIFAR10 is the same as the one on NUS-WIDE, except we do not swap image partitions between the two parties since they are cut from the same images, and thus we assume they have similar discriminative power.

5 Conclusion

We propose Federated Multi-View Training (FedMVT), a semi-supervised learning approach that improves the performance of VFL with limited overlapping samples. FedMVT leverages feature representation estimation and pseudo-labels...
prediction to expand the training set and trains three classifiers jointly to improve models’ representation learning. FedMVT not only significantly improves the performance of the federated model in VFL, but also enhances the performance of the local model of the party that owns labels.

References

[Berthelot et al., 2019] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *CoRR*, abs/1905.02249, 2019.

[Bonawitz et al., 2017] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191, 2017.

[Bousmalis et al., 2016] Konstantinos Bousmalis, George Tzeng, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 343–351. Curran Associates, Inc., 2016.

[Cheng et al., 2019] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjia Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *CoRR*, abs/1901.08755, 2019.

[Chua et al., 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: A real-world web image database from National University of Singapore. In *CIVR*, 2009.

[Clark et al., 2018] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[DLA Piper, 2018] DLA Piper. Data protection laws of the world: Full handbook, 2018.

[Gascon et al., 2016] Adrià Gascón, Philipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptology ePrint Archive*, page 892, 2016.

[GDPR, 2018] GDPR. General data protection regulation, 2018.

[Grandvalet and Bengio, 2004] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Pro-ceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS04, page 529536, Cambridge, MA, USA, 2004. MIT Press.

[Gupta and Raskar, 2018] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *CoRR*, abs/1810.06060, 2018.

[Hardy et al., 2017] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

[Hyun Lee, 2013] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

[Kairouz et al., 2019] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurlien Bellet, Mehdi Bennis, and Arjun Nitin Bhagoji. Advances and open problems in federated learning. 2019.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[Laine and Aila, 2017] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR (Poster)*. OpenReview.net, 2017.

[Li and Wang, 2019] Daliang Li and Junpu Wang. FedNID: Heterogenous federated learning via model distillation. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[Liu et al., 2018] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *CoRR*, abs/1812.03337, 2018.

[McMahan et al., 2016] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[Mohassel and Zhang, 2017] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. *IACR Cryptology ePrint Archive*, page 396, 2017.

[Nock et al., 2018] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Entity resolution and federated learning get a federated resolution. *CoRR*, abs/1803.04035, 2018.

[Oquab et al., 2014] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[Pan et al., 2010] Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760, 2010.

[Peterson et al., 2019] Daniel Peterson, Pallika Kanani, and Viren-dra J. Marathe. Private federated learning with domain adaptation. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[Phong et al., 2018] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shihuo Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[Rivest et al., 1978] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–179, 1978.

[Tarvainen and Valpola, 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1195–1204. Curran Associates, Inc., 2017.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, and Aidan N Gomez. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.
[Vepakomma et al., 2018] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. CoRR, abs/1812.00564, 2018.

[Yang et al., 2019a] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu. Federated Learning. Morgan & Claypool, 2019.

[Yang et al., 2019b] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2):12:1–12:19, 2019.