# Exploring the benefits of differentially private pre-training and fine-tuning for table transformers

***Xilong Wang[1], Pin-Yu Chen[2]***

*[1] Duke University, USA, [2] IBM Research, USA*

Corresponding author: Xilong Wang, xilong.wang@duke.edu

For machine learning with tabular data, a table transformer (TabTransformer) is a state-of-the-art neural network model, while Differential Privacy (DP) is an essential component to ensure data privacy. In this paper, we explore the benefits of combining these two aspects together in the scenario of transfer learning, differentially private pretraining and fine-tuning of TabTransformers with a variety of Parameter-Efficient Fine-Tuning (PEFT) methods, including adapter, LoRA, and prompt tuning. Our extensive experiments on four ACS datasets with different configurations show that these PEFT methods outperform traditional approaches in terms of the accuracy of the downstream task and the number of trainable parameters, thus achieving an improved trade-off among parameter efficiency, privacy, and accuracy.

Keywords: Differential privacy, table transformer, transfer learning

## 1. INTRODUCTION

Table transformer (TabTransformer) [1] is novel deep tabular data modeling for various scenarios, such as supervised and semi-supervised learning. Its main contribution is to transform regular categorical embeddings into contextual ones, thus achieving higher accuracy compared to previous state-of-the-art methods. On the other hand, Differential Privacy (DP) [2] is a frequently used technique to ensure privacy for individual data points in a training dataset. DP-SGD [3], which combines DP with Stochastic Gradient Descent (SGD), is one of the most frequently used optimization techniques in Machine Learning (ML) to train models on sensitive data while safeguarding individual privacy.

In the literature, DP-SGD techniques either fine-tune a pre-trained model or train a model from scratch. However, almost none of them have focused on TabTransformer. In this paper, we implement various recent parameter-efficient fine-tuning techniques, such as LoRA [4], adapter [5], and prompt tuning [6] (both shallow tuning and deep tuning), so as to explore the benefits of differentially private pretraining and fine-tuning for TabTransformers. To summarize, our key contributions are as follows: 1) We study an unexplored scenario for transfer learning in TabTransformers with DP, i.e., implementing various kinds of parameter-efficient techniques in the fine-tuning stage instead of full tuning. 2) Different from previous tabular learning methods which mainly exploited DP at the fine-tuning stage, we study the use of DP-SGD for both pretraining and fine-tuning, thus ensuring end-to-end privacy. 3) Our experiments on four ACS datasets showed that the accuracy outperforms the baselines in most cases, while the parameter efficiency improves by more than **97.86%**. In addition, we report the best advantageous PEFT setting to inform and inspire the future design of DP-aware pretraining and fine-tuning for TabTransformers.

## 2. BACKGROUND AND RELATED WORK

### 2.1 Differential Privacy (DP) and DP-SGD

ML is widely known for its ability to analyze large datasets, identify patterns, and make predictions or decisions based on that data. However, this also introduces the risk of disclosing sensitive information from the training dataset. DP [2] and DP-SGD [3] are introduced to address this issue. A randomized algorithm $\mathcal{A}$ satisfies $(\epsilon, \delta) - DP$ if it holds that:

$$\mathcal{P}[\mathcal{A}(\mathcal{D}) \in S] \leq \exp(\epsilon)\mathcal{P}[\mathcal{A}(\mathcal{D}') \in S] + \delta, \qquad (1)$$

where $\mathcal{P}[\mathcal{A}(\mathcal{D}) \in S]$ is the probability that the output of $\mathcal{A}$ on dataset $\mathcal{D}$ falls within a set $S$, and $\mathcal{P}[\mathcal{A}(\mathcal{D}') \in S]$ is the probability that the output of the $\mathcal{A}$ on a neighboring dataset $\mathcal{D}'$ (which differs from $\mathcal{D}$ by one data point) falls within $S$. The smaller $\epsilon$ is, the stronger privacy guarantee $\mathcal{A}$ has.

Inspired by DP, Differential Privacy Stochastic Gradient Descent (DP-SGD) [3] is one of the most widely used privacy-preserving optimization techniques in ML [7, 8, 9, 10]. It is a two-stage procedure. Formally, given the SGD gradient estimator $g$ evaluated on the training dataset, and define its sensitivity $S_g$ as the maximum of $\left\|g(\mathcal{D}) - g(\mathcal{D}')\right\|_2$. In the first stage, DP-SGD adds a zero-mean Gaussian noise with a given covariance matrix, i.e., $\mathcal{N}(0, S_g^2\sigma^2\mathbf{I})$ to the computed gradient estimator as follows:

$$g(\mathcal{D}) + \mathcal{N}(0, S_g^2\sigma^2\mathbf{I}).$$

In the second stage, DP-SGD passes the gradient estimator through the Clip operator:

$$\text{Clip}(x) = \frac{x}{\max\{1, \|x\|_2 / C\}},$$

so as to fix the sensitivity of the gradient estimator at a hyperparameter $C$. However, regrettably, almost none of DP-SGD techniques have been applied to the study of TabTransformer.

### 2.2 Parameter-Efficient Fine-Tuning (PEFT)

PEFT [11, 12, 13, 5, 4, 6] is an emerging technique in the field of transfer learning that aims to adapt large pretrained models to specific tasks with a smaller number of task-specific parameters. It fine-tunes the pretrained model on a target task while keeping the majority of the original model's parameters frozen. Compared to full-tuning which fine-tunes the entire model, this approach reduces the computational resources and memory requirements needed for task-specific adaptation. PEFT is particularly valuable in scenarios with limited computing resources or when deploying models to resource-constrained environments, without sacrificing task performance. The most popular PEFT techniques include LoRA [4], adapter [5], and (deep/shallow) prompt tuning [6]. Nevertheless, similar to standard ML, PEFT also faces the risk of disclosing sensitive data throughout the fine-tuning procedure and thus needs a privacy guarantee [14].

### 2.3 DP data synthesis

Recent Diffusion Models (DMs) [15, 16, 17, 18, 19] excel at synthesizing high quality images and performing robustly across diverse tasks. Applying Differential Privacy (DP) to diffusion models is an emerging area focused on creating high-fidelity synthetic data while safeguarding individual privacy [20]. Early work on privacy-preserving generative modeling primarily applied DP-SGD [3] to generative adversarial networks (GANs) [21, 22, 23] and variational autoencoders (VAEs) [24]. With the advent of diffusion models [16], researchers began exploring DP-SGD in this setting. Dockhorn, Cao, Vahdat, and Kreis [20] first investigated DP for diffusion, and Ghalebikesabi, Berrada, Gowal, Ktena, Stanforth, Hayes, De, Smith, Wiles, and Balle [25] showed that pre-training on public data before fine-tuning on private data yields state-of-the-art results. Liu, Lyu, Vinaroz, and Park [26] introduced DP-LDM, a latent diffusion model that requires far fewer parameters to fine-tune than pixel-space diffusion. Meanwhile, several custom privacy-focused architectures have emerged, such as DP-MEPF [27] (privatizing feature-embedding means), DPGEN [28] (energy-based modeling with random responses), PrivImage [29] (semantic query functions using public data), and DP-Promise [30] (adding DP noise in early forward steps). Nevertheless, the large size of modern diffusion models still makes fine-tuning computationally expensive, limiting their practical applicability.

Despite extensive research on DP image data synthesis, work on DP tabular data remains limited [31, 32]. PrivSyn [33] was among the first methods to automatically handle general tabular datasets. DP-LLMTGen [34] and SafeSynthDP [35] leverage pretrained Large Language Models (LLMs) for privacy-preserving tabular data generation. Specifically, DP-LLMTGen uses a two-stage fine-tuning process with a novel tabular-focused loss function, then synthesizes data by sampling from the fine-tuned LLM. Meanwhile, the private evolution algorithm, initially designed for image and text data, has been adapted for DP-enabled tabular data generation.

## 3. METHODOLOGY

### 3.1 TabTransformer

TabTransformer [1] is a deep learning architecture for tabular data modeling. It uses contextual embeddings to achieve higher prediction accuracy and better interpretability. It outperforms state-of-the-art deep learning methods for tabular data and is highly robust against missing or noisy data features. The brief structure of TabTransformer is displayed in Fig. 1 (a). The TabTransformer architecture consists of a column embedding layer, a stack of $N$ transformer blocks, and a Multi-Layer Perceptron (MLP). Each transformer layer comprises a multi-head self-attention layer followed by a position-wise feed-forward layer. As shown in Fig. 1 (a), the areas highlighted in red are where we can perform PEFT. To be specific, we implemented LoRA [4] and adapter [5] in transformer blocks, while deep tuning and shallow tuning [6] were exploited in MLP.

### 3.2 Deep tuning and shallow tuning

Visual Prompt Tuning (VPT) [6] is an efficient alternative compared to full fine-tuning for large-scale transformer models. It offers two tuning strategies: VPT-Deep and VPT-Shallow. VPT-Deep prepends a set of learnable parameters to each transformer encoder layer's input, while VPT-Shallow only inserts the prompt parameters to the first layer's input. Inspired by VPT, we proposed deep tuning and shallow tuning, which aims at fine-tuning the MLP of TabTransformer.

### 3.3 Adapter

Adapter [5], as shown in Fig. 2 (b), is a transfer learning approach that allows for efficient parameter sharing and extensibility in large pretrained models. It uses small and task-specific modules that are inserted between the pretrained layers of the base model. These modules have a near-identity initialization and a small number of parameters, which allow for stable training and slow growth of the total model size when more tasks are added.

### 3.4 LoRA

LoRA [4], as shown in Fig. 2 (c), is a low-rank adaptation technique that reduces the number of trainable parameters for downstream tasks while maintaining high model quality. It works by injecting a low-rank adaptation matrix into the pretrained model, which can be shared and used to build many small LoRA modules for different tasks. LoRA makes training more efficient and allows for quick task-switching.

### 3.5 Joint PEFT with DP-SGD

We incorporate DP-SGD into PEFT by initially pretraining a TabTransformer model with DP on a pretraining dataset. Subsequently, we freeze the backbone of the pretrained model and apply the aforementioned PEFT techniques to fine-tune the model in conjunction with DP-SGD on the downstream dataset. This approach serves to safeguard the privacy of both the pretraining dataset and the downstream dataset, thus ensuring end-to-end privacy. To be more detailed, as shown in Fig. 2 (a) and Fig. 2 (c), we combine LoRA with feed forward layer in each transformer block of TabTransformers. Moreover, we inject an adapter between the feed forward layer and the Add & Norm Layer in each transformer block of TabTransformers. For deep tuning and shallow tuning, as shown in Fig. 1 (b), deep tuning tunes certain neurons in every layer of MLP, i.e., the red-marked part in Fig. 1 (b). Meanwhile, shallow tuning only tunes a few neurons in MLP's input layer, i.e., the green-marked part in the figure.

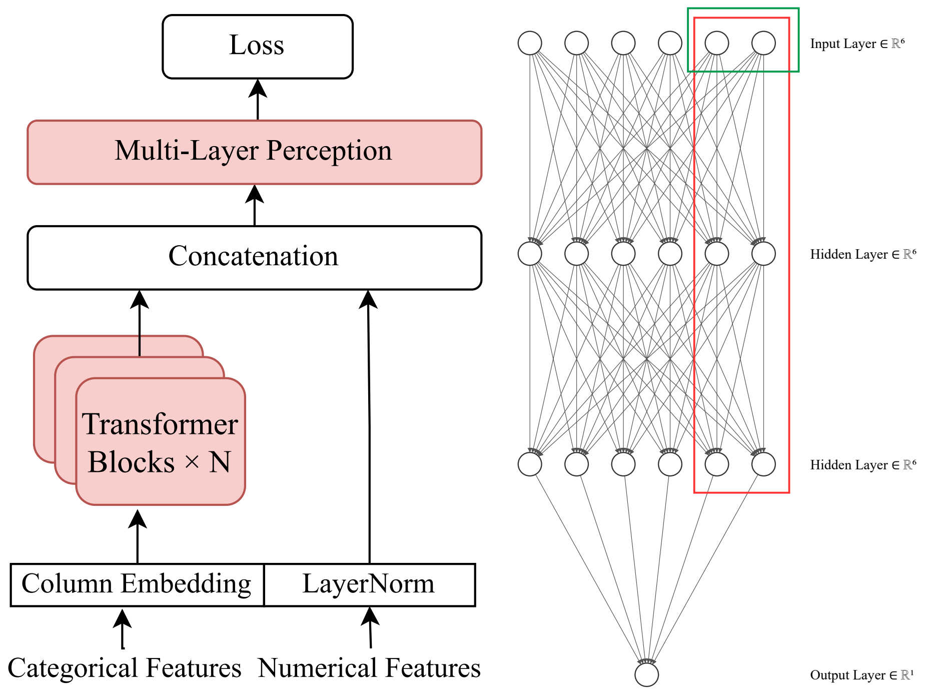
## 4. PERFORMANCE EVALUATION

In this section, we test the performance of all mentioned PEFT approaches and identify the most effective one to benefit future research. For comparison, we exploit two baselines, i.e., full tuning and training from scratch. Furthermore, to illustrate the impact of PEFT, we also evaluate the pretrained model directly on the downstream data without PEFT (i.e., zero-shot Inference). Experimental results clearly show that PEFT methods ensure high parameter efficiency without the loss of accuracy, thus outperforming basic approaches in terms of accuracy, parameter efficiency, and privacy.

### 4.1 Experiment setup

**Datasets:** The ACS dataset [36] is derived from the American Community Survey (ACS) Public Use Microdata Sample (PUMS) data. It gathers detailed demographic, social, economic, and housing data from a representative sample of the U.S. population every year. The ACS covers a wide range of topics, including income, employment, education, housing conditions, and health insurance. By selecting a specific feature from the dataset and converting it into a binary outcome, the ACS dataset provides the following predefined binary prediction tasks:

- **ACS income:** Predict whether an individual's income is above \$50,000, after filtering the ACS PUMS data sample to only include individuals above the age of 16, who reported usual working hours of at least 1 hour per week in the past year, and an income of at least \$100. In our case, the threshold of \$50,000 was



(a) Architecture of TabTransformer.

(b) Overview of Deep Tuning and Shallow Tuning.

**Figure 1** – Genreral framework of parameter-efficient tuning on TabTransformer.

chosen so that this dataset can serve as a comparable replacement to the UCI Adult dataset [37], but the income threshold can be changed easily to define new prediction tasks.

- **ACS public coverage:** Predict whether an individual is covered by public health insurance, after filtering the ACS PUMS data sample to only include individuals under the age of 65, and those with an income of less than \$30,000. This filtering focuses the prediction problem on low-income individuals who are not eligible for Medicare.
- **ACS employment:** Predict whether an individual is employed, after filtering the ACS PUMS data sample to only include individuals between the ages of 16 and 90.
- **ACS travel time:** Predict whether an individual's one-way commute time to work exceeds 30 minutes, after filtering the ACS PUMS data sample to only include individuals between the ages of 16 and 90 who reported commuting to work. This task focuses on understanding factors associated with longer commute durations.

**Distribution shifts:** The original paper of the ACS dataset [36] addresses the issue of distribution shifts between different states in the U.S., evaluating the performance of models, notably Gradient Boosting Machines (GBMs), when trained on data from one state and tested on data from others. The results have demonstrated that distribution shifts can significantly affect model performance, therefore highlighting the importance of adopting transfer learning for applications spanning diverse U.S. states. Inspired by this finding, we explore the use of differentially private pretraining and fine-tuning across different states. Specifically, for each dataset, we chose two states, California (CA) and Indiana (IN), which are geographically distant and have significant differences in population size and economic disparities, exhibiting obvious training set distribution shifts. Thus, we utilized them for the study of pretraining and fine-tuning with TabTransformer. In our setup, all CA samples are used exclusively for pretraining to simulate a transfer learning scenario where the source domain differs from the target domain. The IN samples are then randomly split, with 80% used for fine-tuning and the remaining 20% reserved as the test set to evaluate target domain performance. We do not include CA samples in the test set to ensure that the evaluation strictly measures generalization to

**Table 1** – Detailed information about the ACS dataset.

| Datasets | States | Features | Samples |
|---|---|---|---|
| ACS Income | CA | 10 | 195,665 |
| | IN | 10 | 35,022 |
| ACS Public Coverage | CA | 19 | 138,554 |
| | IN | 19 | 24,330 |
| ACS Employment | CA | 17 | 378,817 |
| | IN | 17 | 67,680 |
| ACS Travel Time | CA | 16 | 172,508 |
| | IN | 16 | 30,932 |

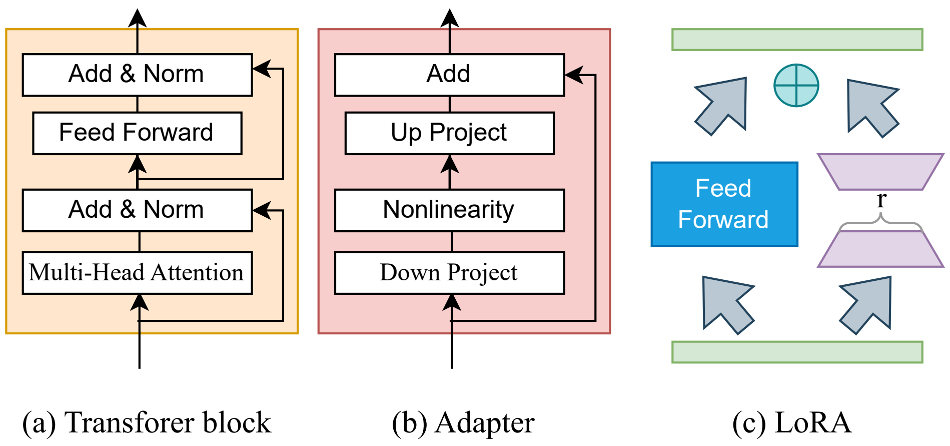


**Figure 2** – PEFT techniques applied on transformer block.

the target distribution (IN). Table 1 provides detailed statistics of the ACS dataset.

**Baselines:** We included the following baselines for evaluation:

- *Full tuning*: In this scenario, after pretraining, full tuning tunes all parameters of the pretrained TabTransformer model.
- *Train from scratch*: This baseline simply trains the entire model on the downstream dataset from scratch without pretraining.
- *Zero-shot inference*: To emphasize the effect of PEFT, after obtaining a pretrained model, this baseline directly evaluates the performance of the same model on the downstream dataset.

**Parameter setups:** Here we specify our parameter settings.

- Let $\epsilon_p$ denote the privacy budget $\epsilon$ used for pretraining and $\epsilon_f$ denote $\epsilon$ used for fine-tuning. The values for $\epsilon_p$ and $\epsilon_f$ are selected from $\{0.5, 1, 2, 4, 8, 16, 32\}$ to cover a wide spectrum of privacy-utility trade-offs, ranging from strong privacy (low $\epsilon$) to weak privacy (high $\epsilon$), consistent with prior work on differentially private machine learning.
- Clipping norm $C = 2$.
- DP parameter $\delta = 10^{-5}$.
- For TabTransformer, we set the hidden (embedding) dimension, the number of transformer blocks, and the number of attention heads to be 32, 4, and 8, respectively. The size of MLP is 5 layers with 72 units for each layer.
- Batch size $\mathcal{B} = 64$ for both pretraining and fine-tuning.
- Full tuning tunes 8 units (tokens) in every MLP layer, and shallow tuning tunes 8 units just in the first layer of MLP.

## 4.2 Experimental results

**Number of trainable parameters.** The degree of parameter efficiency in a PEFT technique hinges on the number of parameters that remain trainable during fine-tuning. Let $N$ represent the number of trainable parameters, and then the $N$ of all the techniques are shown in Table 2.

**Table 2** – Number of trainable parameters of various methods

| Methods | Deep Tuning | Full Tuning | Shallow Tuning |
|---|---|---|---|
| $N$ | 4,408 | 206,193 | 2,072 |
| Methods | Adapter | LoRA | Train from Scratch |
| $N$ | **1,424** | **1,424** | 206,193 |

Based on Table 2, we can arrive at the following conclusion. When we make a comparison between the PEFT methods listed in the table and the baseline methods (full tuning and train from scratch), it becomes evident that all the PEFT approaches have substantially decreased the value of $N$ by at least $\frac{206{,}193-4{,}408}{206{,}193} = \mathbf{97.86\%}$. To delve into the specifics, it's worth highlighting that LoRA and adapter emerge as the most parameter-efficient alternatives, exhibiting a remarkable reduction in $N$ by $\frac{206{,}193-1{,}424}{206{,}193} = \mathbf{99.3\%}$.

**Testing accuracy.** In our endeavor to evaluate the performance of all PEFT methods against the baseline, the TabTransformer model underwent a two-step procedure. Initially, for each dataset, the TabTransformer was subjected to pre-training on the data sourced from California, followed by fine-tuning on the data from Indiana. The entire process of pre-training and fine-tuning is performed using DP-SGD. For evaluation, for each dataset, we randomly split 20% of the data from Indiana as the test set. Furthermore, we opted for the utilization of the accuracy metric, denoted as *Accuracy* (*Acc*), as the primary evaluation criterion for assessing the TabTransformer's ability to predict specific tasks across these d atasets. The detailed results are shown in tables 3 to 6.

**Table 3** – Testing accuracy comparison on ACS income dataset.

| Full tuning | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.4345 | 0.6325 | 0.6985 | 0.6989 | 0.7018 | **0.7168** | 0.7221 |
| 1 | 0.4468 | 0.6811 | 0.7001 | 0.7028 | 0.7031 | 0.7172 | 0.7243 |
| 2 | 0.4714 | 0.7024 | 0.7046 | 0.7054 | 0.7141 | 0.7201 | 0.7302 |
| 4 | 0.5473 | 0.7065 | 0.7075 | 0.7098 | 0.7232 | 0.7313 | 0.7355 |
| 8 | 0.6397 | 0.7088 | 0.7125 | 0.7228 | 0.7253 | 0.7352 | **0.7445** |
| 16 | 0.6859 | 0.7136 | 0.7149 | 0.7255 | 0.7263 | **0.749** | **0.7507** |
| 32 | 0.6889 | 0.7189 | 0.7289 | 0.7348 | 0.735 | **0.7512** | **0.7543** |

| Deep tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6664 | 0.6865 | 0.7012 | 0.7048 | 0.7091 | 0.7109 | **0.7259** |
| 1 | 0.6969 | 0.7099 | 0.7142 | **0.7188** | 0.7195 | **0.7261** | **0.7319** |
| 2 | 0.6999 | 0.7108 | 0.7201 | 0.7212 | 0.7269 | **0.7336** | **0.736** |
| 4 | 0.7001 | **0.7263** | 0.7271 | 0.7285 | **0.7329** | **0.7362** | **0.7385** |
| 8 | 0.7132 | 0.7275 | **0.7319** | **0.735** | **0.7362** | **0.7429** | 0.7432 |
| 16 | 0.7182 | 0.7332 | **0.7359** | **0.7375** | **0.7399** | 0.7438 | 0.7445 |
| 32 | 0.7239 | 0.7362 | 0.7378 | 0.7409 | 0.743 | 0.746 | 0.75 |

| Adapter [5] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | **0.6985** | **0.7086** | **0.7105** | **0.7112** | **0.7131** | 0.7162 | 0.7213 |
| 1 | **0.7044** | **0.7143** | **0.7172** | 0.7185 | **0.7196** | 0.7252 | 0.7259 |
| 2 | **0.7158** | **0.7246** | **0.7248** | **0.7261** | **0.7273** | 0.7288 | 0.7335 |
| 4 | **0.7175** | 0.7261 | **0.7278** | **0.7291** | 0.7298 | 0.7308 | 0.7336 |
| 8 | **0.7209** | **0.7279** | 0.7292 | 0.7335 | 0.7345 | 0.7345 | 0.7366 |
| 16 | **0.7288** | **0.7352** | 0.7358 | 0.7359 | 0.7366 | 0.7376 | 0.7436 |
| 32 | **0.7368** | **0.7386** | **0.7435** | **0.7452** | **0.7453** | 0.747 | 0.7475 |

| LoRA [4] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6754 | 0.6772 | 0.6996 | 0.7008 | 0.7096 | 0.7108 | 0.7116 |
| 1 | 0.6791 | 0.7048 | 0.7113 | 0.7116 | 0.7155 | 0.7182 | 0.7202 |
| 2 | 0.6871 | 0.7155 | 0.7171 | 0.7172 | 0.7181 | 0.7209 | 0.7241 |
| 4 | 0.6966 | 0.7178 | 0.7181 | 0.7196 | 0.7198 | 0.7256 | 0.7256 |
| 8 | 0.7203 | 0.7215 | 0.7216 | 0.7232 | 0.7258 | 0.7266 | 0.7318 |
| 16 | 0.7231 | 0.7253 | 0.7269 | 0.7271 | 0.7292 | 0.7295 | 0.7336 |
| 32 | 0.7272 | 0.7293 | 0.7348 | 0.737 | 0.7373 | 0.7463 | 0.7472 |

| Shallow tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.5019 | 0.5879 | 0.588 | 0.6363 | 0.6801 | 0.6922 | 0.6962 |
| 1 | 0.6348 | 0.6959 | 0.7039 | 0.7094 | 0.7095 | 0.7095 | 0.7143 |
| 2 | 0.6514 | 0.6984 | 0.7098 | 0.7162 | 0.7173 | 0.7175 | 0.7183 |
| 4 | 0.7009 | 0.7046 | 0.7173 | 0.7192 | 0.7203 | 0.7248 | 0.7282 |
| 8 | 0.7142 | 0.7188 | 0.7228 | 0.7275 | 0.7306 | 0.7313 | 0.7333 |
| 16 | 0.7263 | 0.7269 | 0.7272 | 0.7335 | 0.7342 | 0.7388 | 0.7395 |
| 32 | 0.736 | 0.7382 | 0.7392 | 0.7409 | 0.7445 | 0.7449 | 0.7452 |

| Train from scratch | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.633 | 0.6889 | 0.6998 | 0.7002 | 0.7008 | 0.7011 | 0.7099 |

| Zero-shot inference | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.6471 | 0.6604 | 0.6682 | 0.6711 | 0.6814 | 0.7016 | 0.7098 |

**Table 4** – Testing accuracy comparison on ACS travel time dataset.

| Full tuning | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.2835 | 0.3397 | 0.4043 | 0.4696 | 0.4864 | 0.6404 | 0.6492 |
| 1 | 0.431 | 0.5334 | 0.5999 | 0.6355 | 0.6462 | 0.6798 | 0.6828 |
| 2 | 0.4478 | 0.6144 | 0.6565 | 0.6622 | 0.6809 | 0.6851 | 0.6901 |
| 4 | 0.6523 | 0.6828 | 0.6928 | 0.6947 | 0.7039 | 0.7054 | 0.7096 |
| 8 | 0.6989 | 0.7065 | 0.7073 | 0.7146 | 0.7149 | 0.7161 | 0.7169 |
| 16 | 0.7084 | 0.7134 | 0.7157 | 0.7176 | 0.7176 | 0.7195 | 0.7195 |
| 32 | 0.7188 | 0.7191 | 0.7199 | 0.7222 | 0.7233 | 0.7233 | 0.7256 |

| Deep tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.4849 | 0.6129 | 0.7031 | 0.7062 | 0.7092 | 0.7104 | 0.7104 |
| 1 | 0.5609 | 0.6576 | 0.7031 | 0.7126 | 0.713 | 0.7165 | **0.7199** |
| 2 | 0.5969 | 0.7069 | 0.7092 | 0.7153 | 0.7172 | **0.7203** | **0.7203** |
| 4 | 0.679 | 0.7088 | 0.7123 | 0.7195 | 0.7199 | 0.7203 | 0.7226 |
| 8 | 0.6989 | 0.7115 | 0.7176 | 0.7199 | 0.7211 | 0.7222 | 0.7241 |
| 16 | 0.7084 | 0.7119 | 0.7203 | 0.7218 | 0.723 | 0.7233 | 0.7325 |
| 32 | 0.7169 | 0.7184 | 0.7233 | 0.7241 | 0.7249 | 0.7314 | 0.7356 |

| Adapter [5] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | **0.6695** | **0.7069** | **0.7069** | **0.7115** | **0.713** | **0.7184** | **0.7188** |
| 1 | **0.7069** | **0.7115** | **0.7149** | **0.7149** | **0.7157** | **0.7188** | 0.7191 |
| 2 | **0.7073** | **0.7157** | **0.7188** | **0.7191** | 0.7199 | 0.7199 | 0.7218 |
| 4 | 0.7123 | **0.7191** | **0.7191** | **0.7199** | **0.7249** | **0.7272** | **0.7276** |
| 8 | 0.713 | **0.7203** | **0.7207** | **0.7207** | **0.7276** | **0.7295** | **0.731** |
| 16 | 0.7157 | 0.7207 | 0.7245 | 0.7264 | **0.7283** | **0.7314** | **0.7337** |
| 32 | **0.726** | **0.7268** | 0.7279 | 0.7314 | 0.7333 | 0.7363 | 0.7382 |

| LoRA [4] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.418 | 0.4765 | 0.6397 | 0.6733 | 0.6893 | 0.7031 | 0.7054 |
| 1 | 0.6679 | 0.6806 | 0.6893 | 0.7031 | 0.7031 | 0.7092 | 0.7119 |
| 2 | 0.7027 | 0.7046 | 0.7062 | 0.7149 | 0.7149 | 0.718 | 0.7188 |
| 4 | **0.713** | 0.7142 | 0.7149 | 0.7169 | 0.718 | 0.7226 | 0.7253 |
| 8 | **0.7169** | 0.7176 | 0.7184 | 0.7188 | 0.7195 | 0.7245 | 0.7291 |
| 16 | 0.7188 | 0.7191 | 0.7199 | 0.7211 | 0.7226 | 0.7264 | 0.7298 |
| 32 | 0.7226 | 0.7253 | 0.7283 | **0.7337** | **0.7352** | **0.7371** | **0.7421** |

| Shallow tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_p$ \ $\epsilon_f$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.5476 | 0.5984 | 0.6324 | 0.6825 | 0.6901 | 0.6989 | 0.7088 |
| 1 | 0.6855 | 0.6886 | 0.6977 | 0.7008 | 0.7035 | 0.7042 | 0.7092 |
| 2 | 0.7023 | 0.7058 | 0.7069 | 0.7104 | 0.7104 | 0.7115 | 0.7142 |
| 4 | 0.7058 | 0.7081 | 0.7123 | 0.713 | 0.7153 | 0.7172 | 0.7199 |
| 8 | 0.7073 | 0.713 | 0.7165 | 0.7195 | 0.7233 | 0.726 | 0.7291 |
| 16 | **0.7241** | **0.7256** | **0.7264** | **0.7268** | 0.7279 | 0.7291 | 0.7306 |
| 32 | 0.7249 | 0.7256 | **0.7291** | 0.7306 | 0.7321 | 0.7356 | 0.7363 |

| Train from scratch | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.6671 | 0.6791 | 0.6909 | 0.6927 | 0.7055 | 0.7067 | 0.7225 |

| Zero-shot inference | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.6788 | 0.7036 | 0.7071 | 0.7078 | 0.7147 | 0.7162 | 0.7163 |

**Table 5** – Testing accuracy comparison on ACS public coverage dataset.

| Full tuning | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.3352 | 0.6422 | 0.6513 | 0.7051 | 0.7129 | 0.7193 | 0.7281 |
| 1 | 0.6652 | 0.7008 | 0.7244 | 0.7252 | 0.7304 | 0.7351 | 0.7419 |
| 2 | 0.733 | 0.7435 | 0.7513 | 0.7612 | 0.7633 | 0.7659 | 0.7741 |
| 4 | 0.7522 | 0.7616 | 0.7672 | 0.7707 | 0.7717 | 0.7731 | 0.7766 |
| 8 | 0.7647 | 0.7698 | 0.7709 | 0.7723 | 0.7748 | 0.7754 | 0.7778 |
| 16 | 0.768 | 0.7711 | 0.7717 | 0.7768 | 0.7774 | 0.7783 | **0.7824** |
| 32 | **0.7791** | 0.7746 | 0.7776 | 0.7809 | 0.7824 | **0.7871** | **0.7916** |

| Deep tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.7008 | **0.7125** | 0.7127 | 0.7222 | 0.7236 | 0.7283 | 0.7314 |
| 1 | 0.7275 | 0.7291 | 0.732 | 0.7413 | 0.7427 | 0.747 | 0.7513 |
| 2 | 0.7388 | 0.7433 | 0.7454 | 0.7536 | 0.754 | 0.7542 | 0.7585 |
| 4 | 0.7474 | 0.7513 | 0.7536 | 0.7538 | 0.7546 | 0.7645 | 0.7655 |
| 8 | 0.7561 | 0.761 | 0.7616 | 0.7674 | 0.7678 | 0.7686 | 0.775 |
| 16 | 0.7651 | 0.7653 | 0.7672 | 0.7702 | 0.7711 | 0.7721 | 0.7781 |
| 32 | 0.7715 | 0.7727 | 0.7729 | 0.7781 | 0.7785 | 0.7793 | 0.784 |

| Adapter [5] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6626 | 0.6854 | **0.7244** | **0.7351** | **0.7415** | **0.753** | **0.7626** |
| 1 | 0.7213 | 0.7372 | **0.7577** | **0.7616** | **0.7633** | 0.7651 | 0.7674 |
| 2 | 0.7524 | **0.7645** | **0.7661** | **0.7678** | **0.7678** | **0.7754** | **0.7768** |
| 4 | 0.7552 | **0.7682** | **0.7711** | **0.7723** | **0.7758** | **0.7774** | **0.7787** |
| 8 | 0.7635 | 0.7688 | **0.7713** | **0.7725** | **0.7783** | **0.7789** | **0.7797** |
| 16 | 0.767 | **0.7772** | **0.7791** | **0.7793** | **0.7793** | **0.7807** | 0.7815 |
| 32 | 0.7723 | 0.7787 | 0.7809 | 0.7815 | 0.7828 | 0.7838 | 0.7871 |

| LoRA [4] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6815 | 0.7045 | 0.7082 | 0.7154 | 0.7166 | 0.7219 | 0.7464 |
| 1 | 0.7281 | 0.7351 | 0.7388 | 0.7452 | 0.7507 | 0.7509 | 0.7657 |
| 2 | 0.745 | 0.7513 | 0.7522 | 0.7637 | 0.7637 | 0.7641 | 0.7698 |
| 4 | 0.7573 | 0.7643 | 0.7657 | 0.7667 | 0.7672 | 0.7717 | 0.7719 |
| 8 | 0.7649 | 0.7651 | 0.7698 | 0.7707 | 0.7727 | 0.7744 | 0.776 |
| 16 | 0.7682 | 0.7688 | 0.7704 | 0.7739 | 0.776 | 0.7768 | 0.7822 |
| 32 | 0.7719 | **0.7803** | **0.7826** | **0.783** | **0.784** | 0.7853 | 0.79 |

| Shallow tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | **0.7051** | 0.7078 | 0.7125 | 0.717 | 0.7269 | 0.7357 | 0.7548 |
| 1 | **0.7431** | **0.745** | 0.752 | 0.7548 | 0.7559 | **0.7655** | **0.769** |
| 2 | **0.7563** | 0.7563 | 0.7626 | 0.7633 | 0.7678 | 0.7719 | 0.7725 |
| 4 | **0.7663** | 0.7667 | 0.7672 | 0.7684 | 0.7696 | 0.7719 | 0.7741 |
| 8 | **0.7692** | **0.7713** | 0.7713 | 0.7713 | 0.7737 | 0.7746 | 0.7754 |
| 16 | **0.7735** | 0.7739 | 0.7739 | 0.7754 | 0.7776 | 0.7791 | 0.7795 |
| 32 | 0.7739 | 0.7752 | 0.7801 | 0.7805 | 0.7818 | 0.7822 | 0.7822 |

| Train from scratch | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.6196 | 0.7041 | 0.7197 | 0.7261 | 0.7446 | 0.7554 | 0.7653 |

| Zero-shot inference | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.6428 | 0.6774 | 0.7028 | 0.7154 | 0.7279 | 0.7501 | 0.7530 |

**Table 6** – Testing accuracy comparison on ACS employment dataset.

| Full tuning | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6769 | 0.6787 | 0.6804 | 0.7049 | 0.7142 | 0.7145 | 0.7196 |
| 1 | 0.7182 | 0.7284 | 0.7305 | 0.7353 | 0.7355 | 0.7363 | 0.7363 |
| 2 | 0.7357 | 0.7365 | 0.7372 | 0.7373 | 0.7374 | 0.7377 | 0.738 |
| 4 | 0.7376 | 0.7406 | 0.7408 | 0.741 | 0.7417 | 0.7418 | 0.7459 |
| 8 | 0.7382 | 0.7445 | 0.7449 | 0.745 | 0.7457 | 0.7474 | 0.7516 |
| 16 | 0.7387 | 0.7466 | 0.7478 | 0.7486 | 0.7503 | 0.7519 | 0.753 |
| 32 | 0.7446 | 0.7493 | 0.7496 | 0.7527 | 0.7547 | 0.7578 | 0.7581 |

| Deep tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.7074 | 0.715 | 0.7205 | 0.7206 | 0.7274 | 0.7371 | 0.7383 |
| 1 | 0.7265 | 0.7328 | 0.7357 | 0.7368 | 0.7374 | 0.7392 | 0.7402 |
| 2 | 0.7341 | 0.7392 | 0.7401 | 0.7405 | 0.7412 | 0.7433 | 0.7453 |
| 4 | 0.7409 | 0.7417 | 0.7419 | 0.7431 | 0.7443 | 0.7456 | 0.7473 |
| 8 | 0.7428 | 0.7432 | 0.7451 | 0.7469 | 0.747 | 0.7489 | 0.7492 |
| 16 | 0.7435 | **0.7479** | **0.7487** | 0.7492 | 0.7507 | 0.7541 | **0.7584** |
| 32 | **0.751** | **0.7525** | **0.7552** | **0.756** | **0.7581** | **0.7594** | 0.7615 |

| Adapter [5] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | **0.7288** | **0.7335** | **0.7344** | **0.7354** | **0.7391** | **0.7408** | **0.7408** |
| 1 | **0.7379** | **0.7383** | **0.7391** | **0.7402** | **0.7422** | **0.7428** | **0.7435** |
| 2 | **0.7391** | **0.7421** | **0.7421** | 0.7422 | **0.7449** | **0.7455** | 0.7462 |
| 4 | **0.7439** | **0.7443** | 0.7451 | 0.7457 | **0.7476** | **0.7481** | **0.7487** |
| 8 | **0.7443** | **0.7449** | **0.747** | **0.7479** | **0.7487** | **0.7527** | **0.7533** |
| 16 | **0.7464** | 0.7466 | 0.7476 | **0.7501** | **0.7543** | **0.7559** | 0.7564 |
| 32 | 0.7465 | 0.7502 | 0.7522 | 0.7522 | 0.7569 | 0.7572 | **0.7641** |

| LoRA [4] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.6837 | 0.7072 | 0.708 | 0.7181 | 0.7224 | 0.7298 | 0.7312 |
| 1 | 0.7273 | 0.7355 | 0.7359 | 0.7366 | 0.7369 | 0.7391 | 0.742 |
| 2 | 0.7372 | 0.7414 | 0.7419 | 0.742 | 0.7422 | 0.7435 | 0.7452 |
| 4 | 0.7388 | 0.7418 | 0.7424 | 0.7434 | 0.7444 | 0.7449 | 0.7463 |
| 8 | 0.7394 | 0.7425 | 0.7434 | 0.7452 | 0.7463 | 0.7475 | 0.7477 |
| 16 | 0.7443 | 0.7461 | 0.7477 | 0.7484 | 0.7487 | 0.7488 | 0.7501 |
| 32 | 0.7503 | 0.7504 | 0.7505 | 0.7518 | 0.7529 | 0.7547 | 0.7559 |

| Shallow tuning [6] | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon_f$ $\epsilon_p$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| 0.5 | 0.7004 | 0.7091 | 0.7125 | 0.7183 | 0.7199 | 0.721 | 0.7255 |
| 1 | 0.7211 | 0.7228 | 0.7327 | 0.738 | 0.7391 | 0.7392 | 0.7415 |
| 2 | 0.7369 | 0.7398 | 0.7405 | **0.7431** | 0.7439 | 0.7441 | **0.747** |
| 4 | 0.7421 | 0.7433 | **0.7458** | **0.746** | 0.746 | 0.7467 | 0.7478 |
| 8 | 0.7439 | 0.7444 | 0.746 | 0.7467 | 0.7469 | 0.7478 | 0.7513 |
| 16 | 0.7439 | 0.7473 | 0.7477 | 0.7478 | 0.7481 | 0.7513 | 0.7526 |
| 32 | 0.7457 | 0.7481 | 0.7504 | 0.7542 | 0.7568 | 0.7572 | 0.7624 |

| Train from scratch | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.7071 | 0.7191 | 0.7309 | 0.7327 | 0.7401 | 0.7455 | 0.7467 |

| Zero-shot inference | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | 0.5 | 1 | 2 | 4 | 8 | 16 | 32 |
| *Acc* | 0.7088 | 0.7336 | 0.7371 | 0.7447 | 0.7462 | 0.7463 | 0.7478 |

Based on the findings presented in tables 2 to 6, we can infer the following conclusions:

- In all four ACS datasets, all the PEFT methods demonstrate comparable *Acc* when compared to the baselines. For example, in ACS income dataset, when $\epsilon_p, \epsilon_f$ are both set to 32, the *Acc* of deep tuning, adapter, LoRA, and shallow tuning is $0.75, 0.7475, 0.7472,$ and $0.7452$, respectively. Meanwhile, when $\epsilon = 32$, the *Acc* of Train from Scratch and Zero-shot Inference are 0.7099 and 0.7098, respectively. These values suggest that when compared to train from scratch and zero-shot inference, the PEFT techniques increase the *Acc* by at least **4.7%**.
- The *Acc* of PEFT generally surpasses that of full tuning, showcasing significant benefits in terms of privacy, accuracy, and parameter efficiency. To illustrate, in the ACS travel time dataset shown in Table 4, a comparison of all $7 \times 7 = 49$ reported scenarios (representing various combinations of differential privacy parameters $\epsilon_p$ and $\epsilon_f$) between adapter [5] and full tuning shows that adapter consistently outperforms full tuning across the board. Furthermore, in the cases of the ACS income, public coverage, and employment datasets, when considering the same 49 scenarios, adapter surpasses full tuning in 43, 42, and 45 cases, respectively, demonstrating its robustness and effectiveness in a variety of settings.

Hence, to sum up, PEFT techniques achieve excellent levels of accuracy (*Acc*) while demonstrating a remarkably high degree of parameter efficiency. Secondly, PEFT methods exhibit a robust tolerance to low values of $\epsilon$ compared to full tuning. which indicates that PEFT can ensure a higher level of privacy than full tuning. For example, for the ACS income dataset, when $(\epsilon_p, \epsilon_f) = (32, 0.5)$, the *Acc* of deep tuning, adapter, LoRA, and shallow tuning are $0.7239, 0.7368, 0.7272, 0.736$, respectively, while the *Acc* of full tuning is 0.6889. Hence, when $(\epsilon_p, \epsilon_f) = (32, 0.5)$, the *Acc* of PEFT is at least **3.5%** higher than full tuning. For the ACS travel time dataset, when $(\epsilon_p, \epsilon_f) = (0.5, 32)$, the *Acc* of deep tuning, adapter, LoRA, and shallow tuning are $0.7104, 0.7188, 0.7054, 0.7088$, respectively, while the *Acc* of full tuning is 0.6492. Hence, when $(\epsilon_p, \epsilon_f) = (0.5, 32)$, the *Acc* of PEFT is at least **5.62%** higher than full tuning.

Finally, according to tables 3 to 6, it becomes clear that the adapter method surpasses other methodologies in a majority of the scenarios in terms of *Acc* , in 126 out of 196 scenarios adapter performed the best. In addition, adapter is the most parameter-efficient PEFT technique as shown in Table 2. Given these observations, we can confidently conclude that the adapter method offers the most advantageous balance between privacy, accuracy, and parameter efficiency. This superior trade-off highlights its effectiveness in optimizing resource usage while maintaining high levels of data privacy and model accuracy.

## 5. CONCLUSION

In this paper, we presented a pilot study exploring the benefits of combining differentially private pretraining and Parameter-Efficient Fine-Tuning (PEFT) for TabTransformers with a variety of fine-tuning methods, including adapter [5], LoRA [4], deep/shallow tuning [6]. We conducted extensive experiments on four ACS datasets with different configurations. The results in Table 2 indicate that the number of trainable parameters of PEFT techniques reduces at least **97.86%** compared to baselines. The results in tables 3 to 6 show that the accuracy of PEFT methods outperforms baselines in most cases. Hence, compared to three baselines which are either parameter-consuming or ineffective, PEFT techniques achieve a significantly improved trade-off among privacy, accuracy, and parameter efficiency. We also find that adapter is the most optimal setting for PEFT in this setting. Our study uncovers the unexplored benefits and provides new insights into applying PEPT on differentially private pretrained TabTransformer for differentially private transfer learning.

## REFERENCES

[1] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. “Tabtransformer: Tabular data modeling using contextual embeddings”. In: *arXiv preprint arXiv:2012.06678* (2020).

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. “Calibrating noise to sensitivity in private data analysis”. In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer. 2006, pp. 265–284.

[3] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. B. McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. “Deep Learning with Differential Privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021).

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[6] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. “Visual prompt tuning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 709–727.

[7] Muah Kim, Onur Günlü, and Rafael F Schaefer. “Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2650–2654.

[8] Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. “An efficient dp-sgd mechanism for large scale nlu models”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 4118–4122.

[9] Huzaifa Arif, Alex Gittens, and Pin-Yu Chen. "Reprogrammable-FL: Improving Utility-Privacy Tradeoff in Federated Learning via Model Reprogramming". In: *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE. 2023, pp. 197–209.

[10] Yizhe Li, Yu-Lin Tsai, Xuebin Ren, Chia-Mu Yu, and Pin-Yu Chen. "Exploring the Benefits of Visual Prompting in Differential Privacy". In: *arXiv preprint arXiv:2303.12247* (2023).

[11] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 7654–7673.

[12] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. "UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 6253–6264.

[13] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 4582–4597.

[14] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. "Differentially private fine-tuning of language models". In: *arXiv preprint arXiv:2110.06500* (2021).

[15] Prafulla Dhariwal and Alex Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *NeurIPS* abs/2105.05233 (2021).

[16] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *CVPR* (2021), pp. 10674–10685.

[17] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: *ArXiv* abs/2204.06125 (2022).

[18] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. "eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers". In: *ArXiv* abs/2211.01324 (2022).

[19] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *NeurIPS* abs/2205.11487 (2022).

[20] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. "Differentially Private Diffusion Models". In: *TMLR* abs/2210.09929 (2022).

[21] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. "Differentially Private Generative Adversarial Network". In: *ArXiv* abs/1802.06739 (2018).

[22] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. "DP-CGAN: Differentially Private Synthetic Data and Label Generation". In: *CVPR Workshop* (2019), pp. 98–104.

[23] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. "GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators". In: *NeurIPS* abs/2006.08265 (2020).

[24] Bjarne Pfitzner and Bert Arnrich. "DPD-fVAE: Synthetic Data Generation Using Federated Variational Autoencoders With Differentially-Private Decoder". In: *ArXiv* abs/2211.11591 (2022).

[25] Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L. Smith, Olivia Wiles, and Borja Balle. "Differentially Private Diffusion Models Generate Useful Synthetic Images". In: *ArXiv* abs/2302.13861 (2023).

[26] Michael F Liu, Saiyue Lyu, Margarita Vinaroz, and Mijung Park. "Differentially Private Latent Diffusion Models". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856. URL: https://openreview.net/forum?id=AkdQ266kHj.

[27] Fredrik Harder, Milad Jalali Asadabadi, Danica J Sutherland, and Mijung Park. "Pre-trained perceptual features improve differentially private image generation". In: *TMLR* (2022).

[28] Jia-Wei Chen, Chia-Mu Yu, Ching-Chia Kao, Tzai-Wei Pang, and Chun-Shien Lu. "Dpgen: Differentially private generative energy-guided network for natural image synthesis". In: *CVPR*. 2022.

[29] Kecen Li, Chen Gong, Zhixiang Li, Yuzhong Zhao, Xinwen Hou, and Tianhao Wang. *PrivImage: Differentially Private Synthetic Image Generation using Diffusion Models with Semantic-Aware Pretraining*. 2024. arXiv: 2311.12850 [cs.CV].

[30] Haichen Wang, Shuchao Pang, Zhigang Lu, Yihang Rao, Yongbin Zhou, and Minhui Xue. "dp-promise: Differentially Private Diffusion Probabilistic Models for Image Synthesis". In: *USENIX* (2024).

[31] Mengmeng Yang, Chi-Hung Chi, Kwok-Yan Lam, Jie Feng, Taolin Guo, and Wei Ni. *Tabular Data Synthesis with Differential Privacy: A Survey*. 2024. arXiv: 2411.03351 [cs.CR]. URL: https://arxiv.org/abs/2411.03351.

[32] Yuzheng Hu, Fan Wu, Qinbin Li, Yunhui Long, Gonzalo Munilla Garrido, Chang Ge, Bolin Ding, David Forsyth, Bo Li, and Dawn Song. "SoK: Privacy-Preserving Data Synthesis". In: *2024 IEEE Symposium on Security and Privacy (SP)*. 2024, pp. 4696–4713. DOI: 10.1109/SP54263.2024.00002.

[33] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. "PrivSyn: Differentially Private Data Synthesis". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 929–946. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/zhang-zhikun.

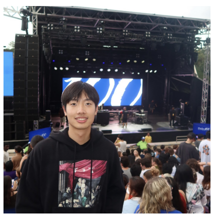
[34] Toan V. Tran and Li Xiong. *Differentially Private Tabular Data Synthesis using Large Language Models*. 2024. arXiv: 2406.01457 [cs.LG]. URL: https://arxiv.org/abs/2406.01457.

[35] Md Mahadi Hasan Nahid and Sadid Bin Hasan. *SafeSynthDP: Leveraging Large Language Models for Privacy-Preserving Synthetic Data Generation Using Differential Privacy*. 2024. arXiv: 2412.20641 [cs.LG]. URL: https://arxiv.org/abs/2412.20641.

[36] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. "Retiring adult: New datasets for fair machine learning". In: *Advances in neural information processing systems* 34 (2021), pp. 6478–6490.

[37] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20. 1996.

## AUTHORS



Xilong Wang is a Ph.D. student in the Department of Electrical and Computer Engineering at Duke University. Prior to that, he earned his bachelor's degree from the University of Science and Technology of China in 2024.

Pin-Yu Chen is a principal research scientist at IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is also the chief scientist of RPI-IBM AI Research Collaboration and PI of ongoing MIT-IBM Watson AI Lab projects. Dr. Chen received his Ph.D. in electrical engineering and computer science from the University of Michigan, Ann Arbor, USA, in 2016.