EAI Endorsed Transactions
on Security and Safety Research Article EAI.EU

Aggregation for Privately Trained Different Types of Local Models

Chunling Han[1,2,*] and Rui Xue[1]

[1]SKLOIS, Institute of Information Engineering, CAS; School of Cyber Security, University of Chinese Academy of Sciences
[2]Indiana University Bloomington

Abstract

Machine learning has been a thriving topic in recent years, with many practical applications and active research aspects. In machine learning, model aggregation is an important area. The idea of model aggregation is to aggregate a global model from trained local models. However, traditional aggregation methods based on parameter averaging can not aggregate models which have different types and structures. Because parameter averaging will fail to average different types of values (parameters). To address this problem, we propose a new aggregation method which will suit for different types of local models. To achieve our goal, we transfer knowledge from local models to the global model. To do so, firstly, we propose differentially private GANs, let local parties generate synthetic data related to their training data. Secondly, we use the majority of prediction votes from local models to label those synthetic samples. Finally, use the labelled synthetic data to train the global model. By combining synthetic data and labels from local models, knowledge can be transferred from local models to the global model. We evaluate our scheme on Adult, MNIST and Fashion MNIST datasets under different settings, experimental results show that our scheme can achieve an accurate global model with low privacy loss. Besides, the easily implemented building blocks make our scheme efficient and practical for applications.

Keywords: Secure aggregation, Generative adversarial network, Differential privacy.

Received on 03 July 2020, accepted on 10 October 2020, published on 13 October 2020

doi: 10.4108/eai.21-6-2021.170237

[*] Corresponding author. Email: hanchun@iu.edu

1 Introduction

Collaboration is an essential factor of success for companies, countries, even globalization of the world. Machine learning sometimes also need collaborations. For example, some medical research institutes wish to collaboratively aggregate a global model to facilitate diagnosis and predictions (for example, the spreading of Covid-19). By aggregating models from different medical research institutes to form a global model, every institute can benefit from others and have a better understanding of the disease. However, sometimes there might be obstacles. These institutes might have different types of models and are not willing to disclose their sensitive datasets.

Why existing works fail. In secure aggregation, most of existing schemes aggregate global models by averaging parameters from local models [18,15,2].

Because these methods are based on the assumption of local models and the global model share exactly the same type and structure, which is indeed a very common situation in many machine learning services. To circumvent private information leaking from local parameters, some works [1,18] release differentially private parameters by adding noise to gradients during training process. However, in fact, averaging local parameters is not always an appropriate solution to model aggregation. Just simply taking the average of local parameters might not directly result in a global model with good accuracy, let alone parameters with noise to achieve differential privacy. Most importantly, local parties might use different types of models, in which parameters can not be averaged. Just like the scenario mentioned above, in such situations, secure aggregation for different types of local models are expected, which motivates our work.

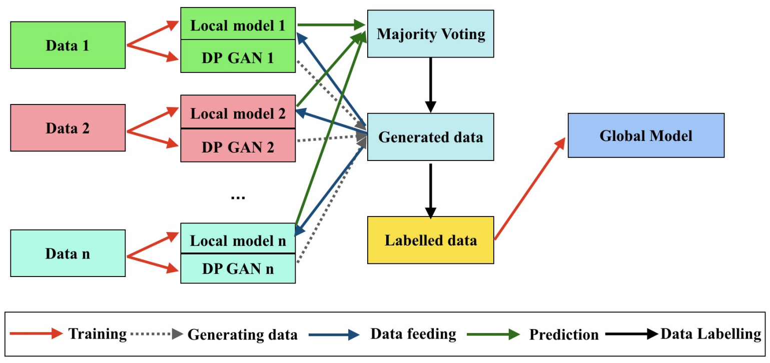


Fig.1. The overview of our aggregation scheme for local models with different types.

To solve the problems, in this paper, we propose an aggregation method to **securely aggregate from different types of local models.** The overview of our approach is shown in Fig.1.

As shown in Fig.1, because local parties use different types of models, we can not simply aggregate from parameters. Instead, we choose to transfer knowledge from local models to the global model. Before using our scheme, we assume local parties have trained their local models based on their privates datasets, and those local models are with different types and structures. Our scheme contains three steps:

- Firstly, besides of their machine learning models, we let local parties train their differentially private local generative adversarial networks (GANs) to generate synthetic data. Then the global party collects the generated synthetic data as unlabelled samples.
- Secondly, with those generated synthetic samples, the global party will require local models' predictions on those unlabelled samples, use the majority voting of local models' prediction results as labels.
- Thirdly, the global model will be trained based on the labelled synthetic data.

In our scheme, we propose a differentially private GAN to generate synthetic data, we clip the loss and then add Gaussian noise to achieve differential privacy.

Because local parties use different types of models, parameter operations will fail. With regard to this limitation, we start from a new perspective, we transfer knowledge from local models to the global model. On one hand, the generated data can reflect statistic distributions of local parties' private datasets. On the other hand, by labelling generated synthetic data, local model can transfer its knowledge learned from its sensitive dataset to the synthetic dataset. Using these two parts of knowledge simultaneously, the global model is supposed to learn knowledge from local parties' private datasets and achieve good accuracy.

We evaluated the performance of our scheme on both text dataset and image datasets, they are Adult, MNIST and Fashion MNIST. We split the dataset for local parties, train different types of local models and GANs on behalf of local parties. We test different Gaussian noise levels, different numbers of local parties. Among those different settings, the global model can achieve 82.81%, 98.28% and 88.94% accuracy for Adult, MNIST and Fashion MNIST respectively. Moreover, our scheme achieves low privacy loss from differentially private GANs. The contributions of our work are as follows:

- We address a new problem that aggregating global model privately from different types of local models.
- In despite of the infeasibility of parameter aggregation, we privately transfer knowledge from local models to the global model by combining differentially private GAN and labelling generated samples from local models .
- We evaluate the performance of our scheme on real-world datasets, experiments show that our scheme can achieve accurate global model with low privacy loss.

The organization of the remainder of this paper is structured as follows: Preliminary is introduced in Section 2. Our scheme along with proof and privacy loss are present in Section 3. Then followed by Evaluations in Section 4. Related works and Conclusion are given in Section 5 and 6. Appendix provides auxiliary proof for our scheme.

2 Preliminary

In this section, we briefly describe the building blocks of our approach, including differential privacy and generative adversary network.

2.1 Differential Privacy

Differential privacy [3,5,6] is a standard for randomized algorithms analyzing dataset providing privacy guarantees. We recall the definition defined in [4].

Definition 1. *A randomized algorithm* M : D → R *with domain* D *and range* R *satisfies* (*ε*,*δ*)*-differential privacy, if for any two adjacent inputs d,d*′ ∈ D *for any subset of outputs S* ⊆ R *it holds that*

$$\Pr[\mathcal{M}(d) \in S] \le e^{\varepsilon}\Pr[\mathcal{M}(d') \in S] + \delta \qquad (1)$$

This definition is based on (*ε*,*δ*)-differential privacy, which means the plain *ε*-differential privacy can be broken with a probability *δ*. In this paper, we use a common value of *δ* as 10^{-5}. Two adjacent datasets means they only differ in a single entry.

To achieve differential privacy for a function, a common method is to add random noise to the function, the magnitude of the noise should be calibrated to the sensitivity of the function.

Definition 2. *For f* : D → R*, adjacent d,d*′ ∈ D*, the* L_2 *sensitivity of f is*

$$\Delta_2 f = \max_{d,d \in \mathcal{D}} \|f(d) - f(d')\|_2 \qquad (2)$$

where $||_2$ means l_2 -norm

The Gaussian noise mechanism achieving differential privacy is defined as follows:

$$\mathcal{M}_f(D) \triangleq f(D) + \mathcal{N}(0, \Delta_2 f^2 \sigma^2) \qquad (3)$$

where $N(0,\Delta_2 f^2\sigma^2)$ is the normal (Gaussian) distribution with mean 0 and standard deviation $\Delta_2 f\sigma$.

2.2 GAN

Generative Adversarial Network (GAN) [7] is a newly invented architecture in machine learning, used for training generative models. The idea of GAN is to let two neural networks compete with each other in a game, one learns to generate synthetic data while the other one learns to distinguish between real and synthetic data. Given a training set, the outcome of a GAN is a generative network that can generate new data with the same statistics as the training set.

Generative Adversarial Networks (GAN) consists of two models: a generator G and a discriminator D. Generator G takes random noise $z \sim p_z(z)$ as input, tries to output synthetic samples of data with distribution approximates real data's distribution $x \sim p_{data}(x)$. The discriminator D will estimate the probability that a sample is a real data comes from the training dataset rather than a synthetic data generated from G. These two models are simultaneously trained in a competitive way, the goal of GAN is training G and D playing a two-player minmax game with the value function V(G,D):

$$\min_G \max_D V(G,D) = E_{x \sim p_{data}(x)}[\log(D(x))] + E_{z \sim p_z(z)}[1 - \log(D(G(z)))] \qquad (4)$$

3 Our Approach

In this section, we illustrate our scheme for secure aggregation from different types of local models. Notice that, our secure aggregation method can also generalize traditional model aggregation where local models share the same model structure.

3.1 Roles of participates

There are two roles in our scheme, local parties and the global party. Local party possesses its sensitive dataset and every local party develops its own machine learning model privately. The global party is in charge of aggregating a global model from local machine learning models.

In our scheme, we consider an honest but curious global party, which will participate in the system honestly but always wants to steal privacy information from local parties.

Local parties are also honest but curious, they participate in the system honestly, and also want to steal sensitive information from other parties. They might collude with each other but would not destroy their collaboration of aggregation.

3.2 Train local models and GANs

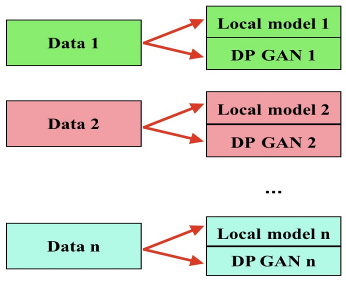


Fig.2. Local parties train their own machine learning models and GANs privately.

As mentioned before, our method can suit for local models with different types. As shown in Fig.2, we use different colors to represent different local models. Local parties train their own machine learning models and develop differentially private GANs.

The types and structures of machine learning models differ among local parties. For example, in our experiment, there are five different machine learning models used among local parties for Adult dataset, they are Logistic Regression, Random Forest, SVM (support vector machine), Decision Tree and Neural Net-work. For MNIST and Fashion MNIST datasets, local parties use different CNN (convolutional neural network) structures, they are different in the number of layers, number of parameters, etc.

Local parties can also use different types and structures of GANs (not mandatory). For examples, in our experiment, we use three different types of GANs: traditional GAN, DCGAN and Variational Autoencoder. Local parties train their GANs based on their own sensitive datasets privately. To prevent generated data repeating or reflecting sensitive features of the training data, we introduce differentially private GAN structure, described in Algorithm 1.

Algorithm 1 Differentially Private GAN

```
Input: Discriminator $D(\theta)$, generator $G(\phi)$, real dataset $\{x_1, x_2, x_3, ..., x_N\}$, loss function $D(\theta)$, loss norm bound C, Gaussian noise level $\sigma$.
Parameter: Initialize $\theta$ and $\phi$.
 1: for epoch $t$ in range $(0, T)$ do
 2:     Generator generates a batch of synthetic data
 3:     $S_t \leftarrow G(z), z \sim p(z)$, with batch size as $B$.
 4:     Train the discriminator D
 5:     Select a batch of real data $B_t$, with batch size $B$.
 6:     The discriminator predicts on the synthetic data $S_t$ and real data $B_t$, calculate
        the loss
 7:     For $s_i \in S_t$, the loss $D\_syn_i = D(\theta, s_i)$.
 8:     For $x_i \in B_t$, the loss $D\_real_i = D(\theta, x_i)$.
 9:     Clip loss
10:     $D\_real_i \leftarrow D\_real_i / \max(1, \frac{||D\_real_i||_2}{C})$.
11:     Add noise
12:     $D\_real_t \leftarrow \frac{1}{B}(\sum D\_real_i + \mathcal{N}(0, C^2\sigma^2))$.
13:     Compute loss
14:     $D\_syn_t = \frac{1}{B}\sum D\_syn_i$
15:     $D\_loss_t = \frac{1}{2}(D\_real_t + D\_syn_t)$
16:     The discriminator $D$ computes the gradient according to $D\_loss_t$ and updates
        its parameters $\theta$, then set the discriminator $D$ untrainable.
17:     Generate synthetic data
18:     The generator $G$ generates a batch of synthetic data from noise $z$. $S'_t \leftarrow G(z), z \sim$
        $p(z)$, the size of $S'_t$ is $S$.
19:     The discriminator $D$ predicts on the synthetic data $S'_t$, compute the loss
20:     $g\_loss_t = \frac{1}{S}\sum D(\theta, s'_i)$
21:     Compute gradient
22:     The discriminator $D$ and the generator $G$ combine as a GAN, then $D$ calculates
        the gradient according to $g\_loss_t$ for the GAN.
23:     Generator updates parameters
24:     The generator updates its parameters according to the gradient.
25: end for
    Output: Differentially private GAN, compute the overall privacy loss $(\varepsilon, \delta)$ using
    moments accountant [1].
```

The perspective of Algorithm 1 is that even though generator has no access to training dataset, it has access to discriminator's parameters, which might have encoded sensitive features of training data. To prevent generator from stealing sensitive information from discriminator's parameters, reflecting them to generated synthetic samples, we need to make sure the derivatives from discriminator's parameters to generator's parameters contain no sensitive features. In Algorithm 1, we provide a differentially private parameter update for generator.

As shown in Algorithm 1, we need the following steps in every epoch of GAN training:

- Train the discriminator on a batch of synthetic data from generator and a batch of real data from training dataset, compute loss on every sample.
- For loss computed on real data, clip loss norm to a threshold C, i.e., if $||D\_real_i||_2 \leq C$ then D_real_i would be preserved, if $||D\_real_i||_2 \geq C$, then D_real_i would be clipped to be norm of C.
- Sum up the loss D_real_i and then add Gaussian noise (with mean 0 and standard variance σ) to the sum of the clipped loss.
- Compute the whole loss D_loss_t, use the loss to compute gradients, update discriminator's parameters. Then set the discriminator untrainable.
- Generator G generates a batch of synthetic data, the discriminator D predicts on the synthetic data, computes the loss and then computes the gradients.
- Generator G updates its parameters according to the gradients.

In Algorithm 1, we use Gaussian differential privacy mechanism to achieve a differentially private discriminator. Because the discriminator is differentially private, through propagation, the generator and the generated synthetic samples from the generator are also differentially private.

3.3 Labelling generated synthetic data

Every local party generates an amount of synthetic samples for the global party. The global party collects synthetic samples from every local party, then submits them to all local models to get predictions. We use majority voting of prediction results as labels for the generated synthetic data.

The idea of majority voting is to select the largest count of the vote to ensemble the decision. Let m be the number of classes in our task, n be the number of local parties. The label count (for a given class $j \in [m]$ and an input x) is the number of local parties assigned class j to input x: $n_j(x) = |\{i : i \in [n], f_i(x) = j\}|$. For an input x, the ensemble prediction j will be the class with the largest count:

$$j = \operatorname{argmax}\{n_j(\boldsymbol{x})\} \quad (5)$$

Notice that, the shortcoming of this method is the majority voting result will be sensitive to individual votes, especially when the votes for two categories are equal. If one individual reversed its prediction result, the ensemble prediction would be reversed as well.

To overcome this problem, we also consider the second majority prediction class count. If the first majority and the second majority votes are equal or only with distance 1, we remove the related generated sample. The reasons for the equality and closeness of the first and second majority probably come from the corresponding synthetic sample being of poor quality or errors and compromised behaviors of local parties.

After obtaining those labelled synthetic samples, the global party will use them to train the global model.

3.4 Differential privacy proof and privacy loss

In this subsection, we prove Algorithm 1 in our scheme can be bounded as (ε,δ)differentially private.

Privacy loss is a random variable dependents on the random noise added to the algorithm. A mechanism M is (ε,δ)-differentially private is equivalent to a certain tail bound on M's privacy loss random variable. In this paper, we use the moments accountant introduced in [1] to keep track of a bound on the moments of the privacy loss random variable (Equation (6)). The moments accountant can be applied for composing Gaussian mechanisms with random sampling. As shown in Algorithm 1, we update the state by sequentially applying Gaussian differentially private mechanisms during training the discriminator.

3.5 Differential privacy proof

Proof. We compute the log moments of the privacy loss random variable, which compose linearly. Then combine the moments bounds with the standard Markov inequality to obtain the tail bound, which is the privacy loss in the sense of differentially privacy.

For neighboring databases $d,d' \in D^n$, a mechanism M, auxiliary input aux, and an outcome $o \in R$, define the privacy loss at o as:

$$c(o; \mathcal{M}, \text{aux}, d, d') \triangleq \log \frac{\Pr[\mathcal{M}(\text{aux}, d) = o]}{\Pr[\mathcal{M}(\text{aux}, d') = o]} \quad (6)$$

As shown in Equation (6), this is an instance of adaptive composition, which we let the auxiliary input aux of the kth mechanism M_k be the output of all the previous mechanisms.

For a given mechanism denoted as M, we define the λth moment $\alpha_M(\lambda;\text{aux},d,d')$ as the log of the moment generating function evaluated at the value λ:

$$\alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d') \triangleq \log \mathbb{E}_{o \sim \mathcal{M}(\text{aux},d)}[\exp(\lambda c(o; \mathcal{M}, \text{aux}, d, d'))]$$

Then we define bound for all possible $\alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d')$ as:

$$\alpha_{\mathcal{M}}(\lambda) \triangleq \max_{\text{aux},d,d'} \alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d') \quad (7)$$

Which can be used to prove privacy guarantees of a mechanism. The maximum is calculate over all possible aux and all the neighboring databases d,d'.

We recall and use some properties of α introduced and proved in [1] to prove the bound.

Theorem 1. *Let α_M defined as above, then*

1. *[Composability] Suppose that a mechanism* M *consists of a sequence of adaptive mechanisms* $M_1,\ldots,M_k$, *where* $\mathcal{M}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$. *Then, for any λ,*

$$\alpha_{\mathcal{M}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda) \quad (8)$$

2. *[Tail bound] For any $\varepsilon > 0$, the mechanism* M *is (ε,δ)-differentially private (we use a common value for δ as 10^{-5}) for*

$$\delta = \min_{\lambda} \{\exp(\alpha_{\mathcal{M}}(\lambda) - \lambda\varepsilon)\} \quad (9)$$

Now we need to prove the bound for every step value $\alpha_{M_t}(\lambda)$. To compute the bound, we need some parameters as follows:

- We use $f(\cdot)$ to denote the function applying Gaussian differentially private mechanism, which is the loss $D\_$ real_i in Algorithm 1, we clip the loss function $f(\cdot)$ to a threshold C (set as 1), therefore we have $||f(\cdot)||_2 \leq C$.

- σ is the standard variance of Gaussian distribution, in Algorithm 1, $\sigma \geq 1$
- Sampling probability from the training dataset is denoted as q, in Algorithm 1, the sampling probability is B/N, in which B is the batch size of selected real samples; N is the total number of samples of training dataset. – Training epochs (steps) in Algorithm 1 is T.

With these settings, we can obtain a bound for $\alpha_M(\lambda)$, we use a theorem in work [1] to facilitate our proof.

Theorem 2. *Suppose that* $f: D \rightarrow R^p$, *with* $||f(\cdot)||_2 \leq C$, $C = 1$. *Let* $\sigma \geq 1$ *and let* J *be a sample from* $[n]$, *where each* $i \in [n]$ *is chosen independently with probability* $q < \frac{1}{16\sigma}$. *Then for any positive integer* $\lambda \leq \sigma^2 \ln \frac{1}{q\sigma}$, *the mechanism*

$M_t(d) = \sum_{i \in J} f(d_i) + N(0, C^2\sigma^2)$ *satisfies*

$$\alpha_{\mathcal{M}_t}(\lambda) \leq \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3) \quad (10)$$

The proof for Theorem 2 is attached in Appendix. With T steps composed, we can obtain a whole bound for Algorithm 1.

$$\alpha_{\mathcal{M}}(\lambda) \leq T\left(\frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3)\right) \quad (11)$$

3.6 Privacy loss

With the bound for $\alpha_M(\lambda)$, we can derive a measurement of the differential privacy parameters (ε,δ) for Algorithm 1. With $\lambda \leq \sigma^2 \ln(1/q\sigma)$, $q < \frac{1}{16\sigma}$,

$$\alpha_{\mathcal{M}}(\lambda) \leq \frac{Tq^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}\left(\frac{Tq^2\lambda^2\frac{q\lambda}{\sigma}}{\sigma^2}\right) \quad (12)$$

We can see the bound for $\alpha_M(\lambda)$ is dominated by $\frac{Tq^2\lambda(\lambda+1)}{(1-q)\sigma^2}$. We set a slightly tight bound for $\alpha_M(\lambda)$ for easy forward reduction as:

$$\alpha_{\mathcal{M}}(\lambda) \leq Tq^2\lambda^2/\sigma^2 \quad (13)$$

Then combine with Theorem 1.1 [Tail bound], for any $\varepsilon < Tq^2$, we can have:

$$Tq^2\lambda^2/\sigma^2 \leq \lambda\varepsilon/2 \quad (14)$$

$$\exp(-\lambda\varepsilon/2) \leq \delta \quad (15)$$

According to these two inequality, and $\lambda > 1$, $\sigma > 1$, we can have:

$$Tq^2\lambda\varepsilon/\sigma^2 \leq \varepsilon^2/2 \quad (16)$$

$$\lambda\varepsilon \geq 2\ln\frac{1}{\delta} \quad (17)$$

Then we set the bound for ε, which is the relation between σ and ε.

$$\varepsilon = 2q\sqrt{T\ln(\frac{1}{\delta})}/\sigma \quad (18)$$

With this evaluation method for ε, we can evaluate the bound (ε,δ) and claim our Algorithm 1 is (ε,δ)-differentially private.

4 Evaluation

4.1 Datasets

We evaluate the performance of our scheme on Adult, MNIST and Fashion MNIST datasets. The Adult dataset is a collection of census data with 48,842 examples, every example has 14 attributes. It is a binary classification and its prediction task is to determine whether a person makes over 50K a year [10]. MNIST is a 10-class handwritten digit recognition dataset consisting of 60,000 training examples and 10,000 test examples [11], each example is a 28 × 28 size greyscale image. Similarly, Fashion MNIST is a 10-class dataset of fashion images, also consisting of 60,000 training examples and 10,000 testing examples [19], each example is a 28 × 28 size gray-level image.

With Adult, MNIST and Fashion MNIST, our scheme can be evaluated broadly on different types of datasets. Adult dataset is a text and number based dataset (produced in 1996). MNIST (produced in 1998) is an image based dataset and has been as a benchmark for machine learning and data science algorithms for years, and now Fashion MNIST (produced in 2017) serves as an alternative replacement for the original MNIST dataset benchmarking machine learning algorithms.

4.2 Train local machine learning models and DP-GANs

In our experiments, we let local parties develop different machine learning models, which have different types and structures. We develop Logistic Regression, Random Forest, SVM (support vector machine), Decision Tree and Neural Network among local parties for Adult. For MNIST and Fashion MNIST, because they are image based datasets, we deploy different CNN (convolutional neural network) structures. CNNs can achieve good accuracies for image recognition tasks.

We split the training dataset among local parties and train every local machine learning model on their part of training dataset, test local models' performance on test dataset, then take the average of local models' accuracies as **"standalone"** accuracy for local models.

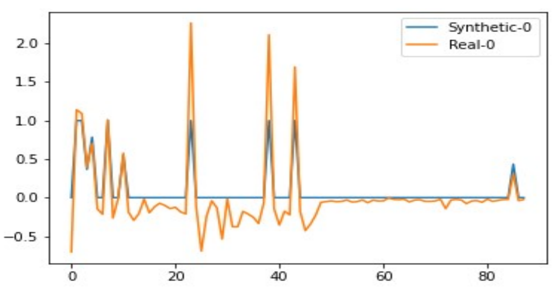
Next, local parties develop their GANs to generate synthetic samples. We also develop different GAN structures, which is not mandatory in our scheme. For Adult dataset, we use traditional GANs and Variational Autoencoders (VAEs) among local parties. For MNIST and Fashion MNIST, we use traditional GANs and DCGANs. VAE and DCGAN are GAN variants, DCGAN [16] uses deep convolutional neural network in GAN structure, while VAE [9,17] uses variational encoder and decoder as GAN structure.

We develop differentially private GANs according to Algorithm 1. It turns out that choosing loss function as target of applying differential privacy benefits our experiments. It is easy to program and suitable for different GAN structures. Also, there is no need to modify core functions in Tensorflow or Keras to implement our scheme. All our experiments are programmed in Python and executed on Google Colab which provides free access to GPUs and has libraries such as Keras, TensorFlow embeded. We will open source our codes along with this paper.

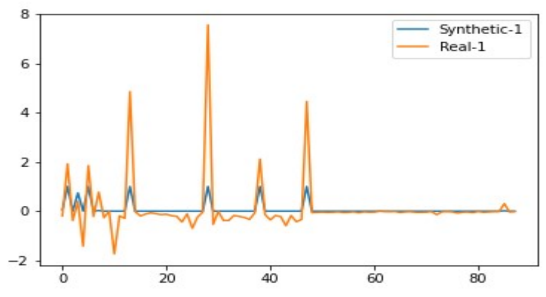
In Fig.4, we show some generated synthetic samples for Adult, MNIST and Fashion MNIST with 5 local parties, loss clipping threshold $C = 1$ and noise level $\sigma =$ 1.0. For Adult, the generated synthetic samples are from a local party using VAE. Generated synthetic MNIST samples are from a traditional GAN and generated synthetic Fashion MNIST samples are from a local party using DCGAN. Because Adult dataset is number and text based dataset, we process samples and standardize each sample to 88 numerical features, then feed the processed samples to GANs. Therefore, in Fig.4 (a) (b), we show samples in 88 features instead of using the original form of data.

As shown in Fig.4, the synthetic samples generated from differentially private GANs look similar with real samples. Because we add Guassian noise to GANs to achieve differential privacy, some generated synthetic samples are a little bit blurry.

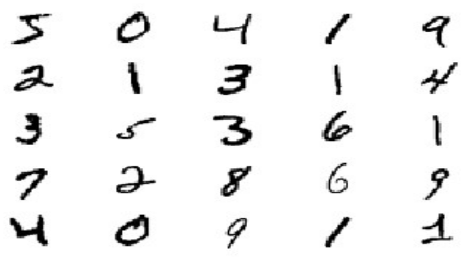
After training local GANs, local parties are ready to generate synthetic samples. We let the number of generated synthetic samples from every local party be the same size with local party's training dataset. The global party collects generated synthetic samples from local parties, submit the synthetic samples to local models. Local models predict on those samples. Then the global party collects the prediction results and uses majority voting to ensemble labels. If the first majority voting is equal or only has one count more than the second majority voting count, the related samples will be removed by the global party.

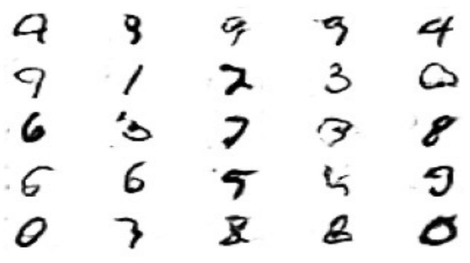


(a) Real and synthetic class -0 samples.



(b) Real and synthetic class -1 samples.





(c) Real MNIST samples. (d) Synthetic MNIST samples.

(e) Real Fashion MNIST samples.

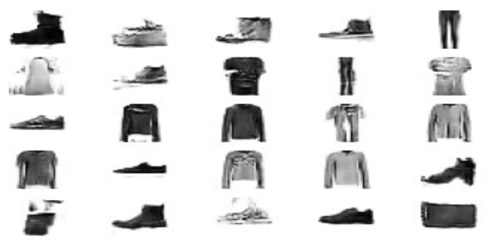

(f) Synthetic Fashion MNIST samples.

Fig.4. Real and Synthetic (a)(b) samples of Adult, synthetic samples are from a VAE; Real (c) and synthetic (d) samples of MNIST, synthetic samples are from a traditional GAN; Real (e) and synthetic (f) samples of Fashion MNIST, synthetic samples are from a DCGAN.

Next, global models will be trained on those labelled generated synthetic datasets. For Adult dataset, the global model is a 4-layer neural network. For MNIST and Fashion MNIST, global models are CNN structures. After training global models for different datasets, we test global models on real test datasets from Adult, MNIST and Fashion MNIST to obtain global accuracies. In Table 1, we list the accuracies of global models for different datasets, we also list the corresponding baseline accuracies which come from machine learning models trained on the whole real training datasets, and the standalone accuracies which are the average of local models' accuracies.

As shown in Table 1, the global models aggregated by our scheme achieve higher accuracies than local model's standalone accuracies, indicating our scheme can provide privacy protection and achieve accurate global models simultaneously. On the other hand, because we use differential privacy in our scheme, global models achieve slightly lower accuracies than baselines. This inferior performance mainly due to two reasons: firstly, the global model is trained on generated samples not on real samples, but tested on real samples, this gap causes some decline of accuracy; secondly, we add noise to GAN to achieve differential privacy, the added noise will to some extent affect the quality of generated synthetic samples, thus affecting the accuracy of trained global model.

Table 1. Accuracy of Global model, Baseline and Standalone on different datasets.

Dataset	Adult	MNIST	Fashion MNIST
Baseline	0.8474	0.9920	0.9250
Standalone	0.8246	0.9816	0.8760
Global	**0.8281**	**0.9828**	**0.8894**

We also evaluate our scheme under different Gaussian noise levels σ. We also compute the privacy loss for every local party according to different noise levels. With noise level σ and δ (we set δ as a common value 10^{-5}), we can calculate the privacy loss ε according to Equation (18). Some other parameters are as follows: batch size B for Adult, MNIST and Fashion MNIST are 256, 128 and 128 respectively. Training epochs T for Adult, MNIST, Fashion MNIST are 1,000, 10,000, and 10,000. Total numbers of training samples N in every local party for Adult, MNIST, Fashion MNIST are 7,814, 12,000 and 12,000. We list the related experimental results in Table 2.

As seen from Table 2, with more noise added, we can achieve stronger privacy protection which means lower privacy loss. Due to more noise is added to differentially private GANs, the generated samples will be affected, then result in less accurate global models.

To compare global models' accuracies under different noise levels with baseline and standalone accuracies, we plot the comparison in Fig.6

As shown in Fig.6, when the noise level is low, the global models achieved by our scheme can exceed local model's standalone accuracies. With more noise added to obtain stronger privacy protection, the global models' accuracies will be slightly lower than local model's standalone accuracies. In fact, this tradeoff between privacy and performance is inevitable in many other privacy-preserving schemes as well. In our scheme, the global party can alter the noise level to balance between privacy loss and accuracies.

Table 2. Global models' accuracies and privacy loss under different noise levels.

Dataset	Noise level	Privacy loss	Accuracy
Adult	$\sigma = 1.0$	$(\varepsilon,\delta) = (7.0306,10^{-5})$	0.8281
	$\sigma = 1.5$	$(\varepsilon,\delta) = (4.6870,10^{-5})$	0.8236
	$\sigma = 3.0$	$(\varepsilon,\delta) = (2.3435,10^{-5})$	0.8147
	$\sigma = 5.0$	$(\varepsilon,\delta) = (1.4061,10^{-5})$	0.8109
MNIST	$\sigma = 1.0$	$(\varepsilon,\delta) = (7.2385,10^{-5})$	0.9828
	$\sigma = 1.5$	$(\varepsilon,\delta) = (4.8257,10^{-5})$	0.9819
	$\sigma = 3.0$	$(\varepsilon,\delta) = (2.4128,10^{-5})$	0.9789
	$\sigma = 5.0$	$(\varepsilon,\delta) = (1.4477,10^{-5})$	0.9677
Fashion MNIST	$\sigma = 1.0$	$(\varepsilon,\delta) = (7.2385,10^{-5})$	0.8894
	$\sigma = 1.5$	$(\varepsilon,\delta) = (4.8257,10^{-5})$	0.8779
	$\sigma = 3.0$	$(\varepsilon,\delta) = (2.4128,10^{-5})$	0.8702
	$\sigma = 5.0$	$(\varepsilon,\delta) = (1.4477,10^{-5})$	0.8647

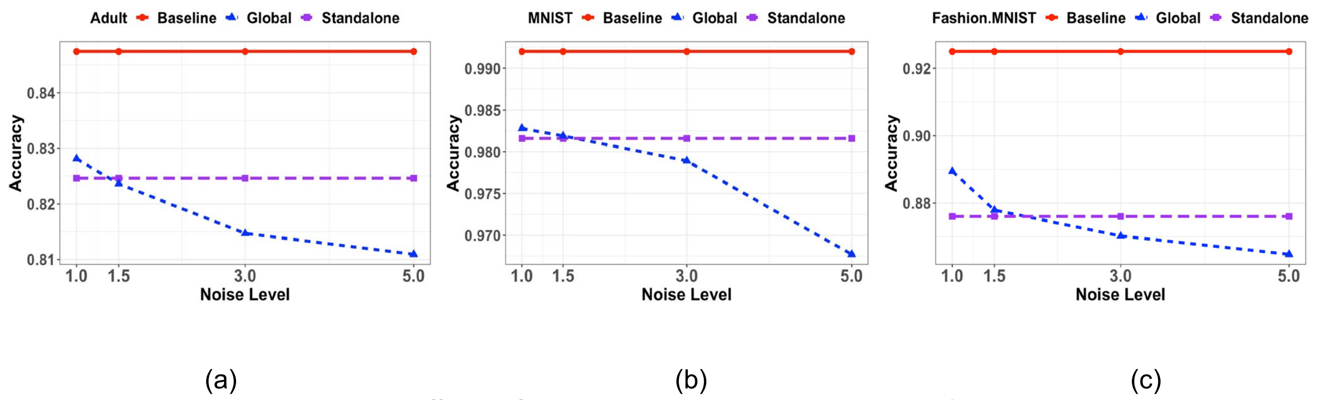
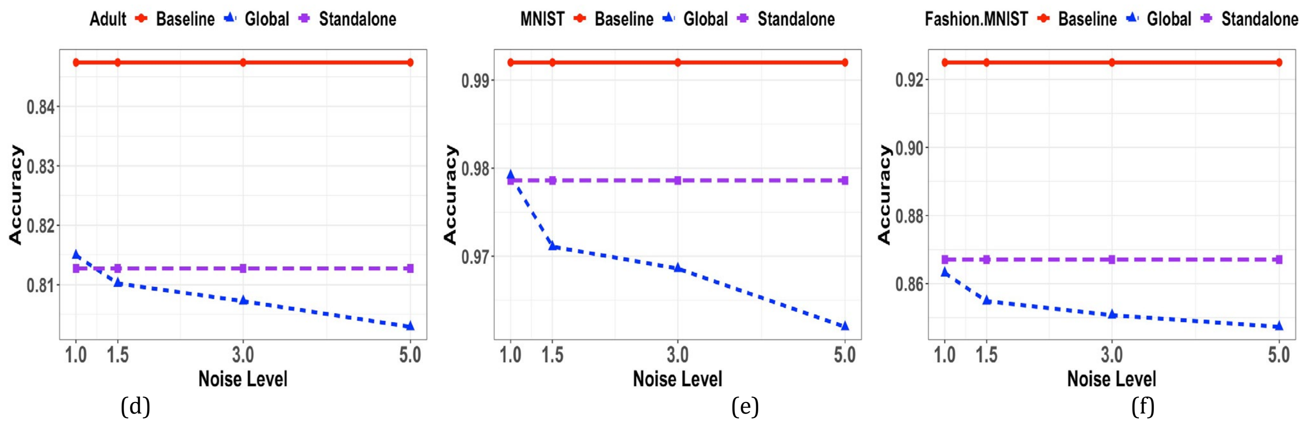


Fig.6. With 5 local parties and different Gaussian noise levels, global models' accuracies, baseline and standalone accuracies for Adult, MNIST and Fashion MNIST.

We also test our scheme under different numbers of local parties. In Fig.8, we plot global models accuracies aggregated from 10 and 20 local models along with local models' standalone and the baseline accuracies.

Notice that, because we split the training data evenly among local parties, with more local parties involved, every local party will have less training data and achieve less accurate local models. Consequently, with less data, the quality of generated samples from local differentially private GAN will decrease, therefore resulting in less accurate global models trained on those generated samples.

In conclusion, according to those experiments, our scheme can achieve accurate global models from different types of local models, meanwhile, providing satisfying privacy protection with low privacy loss.



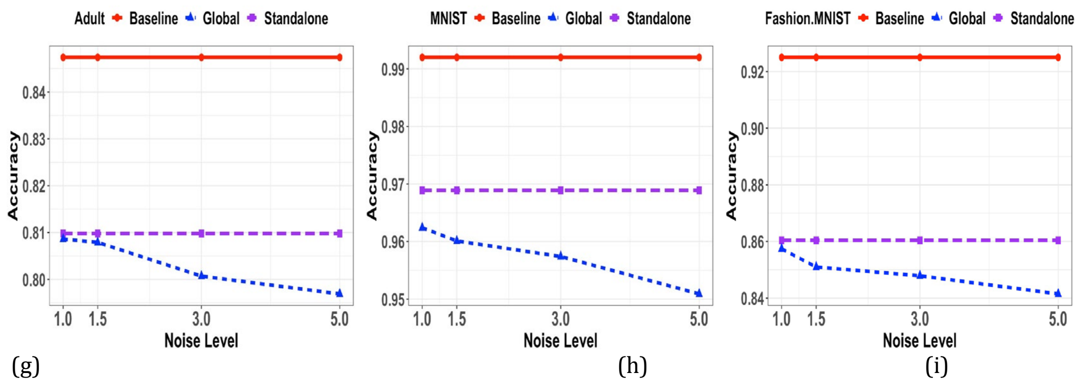


Fig.8. Under different Gaussian noise levels, ((d)(e)(f) from 10 local parties and (g)(h)(i) from 20 local parties), global models' accuracies, baseline and standalone accuracies for Adult, MNIST and Fashion MNIST.

4.3 Comparison with other works

We compare our scheme with two state-of-the-art related works [18,12] about secure machine learning model aggregation. DP-DSSGD [18] is designed for aggregating local models with the same structure. Local models select and upload small part of differentially private parameters to the global party, then the global party uses the average of local models' parameters to form global model's parameters. SecureML [12] uses secure two-party computation to train a global model. They let data owners distribute their private data among two non-colluding servers who train various models on the joint data using secure two-party computation (2PC). We compare our scheme with these two schemes evaluated on MNIST dataset (Due to lack of experimental results from these two works, we only compare the performance on MNIST), shown in Table 3.

Table 3. Comparison among our scheme and two related works.

Scheme	Methods	Versatility	Accuracy
DP-DSSGD [18]	DP, Selective SGD	**Same type**, **deep learning**	97.00%
SecureML [12]	2PC	**Different types**, **machine learning**	93.10%
Our scheme	DP-GAN	**Different types**, **machine learning**	**98.26%**

Because DP-DSSGD computes privacy loss based on every parameter, their privacy loss will be huge when local parties upload many parameters. With 0.1 portion of parameters uploaded from local models, their privacy loss can still be larger than our scheme. With 0.1 portion of parameters uploaded, DPDSSGD achieves lower accuracy than our scheme. Besides, our scheme can suit for different types of machine learning models while DP-DSSGD can only suit for same type of deep learning models.

SecureML uses Garbled Circuits (a popular tool for secure two-party computation) to learn a global model without knowing data owners' private dataset. Because they need to compute almost the entire machine learning process under Garbled Circuits, which are only suitable for boolean circuits, they need to modify non-linear activation functions, thus causing decline of global model's accuracy, they only obtain 93.1% on MNIST. Even worse, Garbled Circuits can be extremely time-consuming and sometimes space-consuming, which is very inefficient for machine learning algorithms.

5 Other Related Works

There are some works originally designed for differentially private deep learning can bring inspirations to our work. Through modifications, some ideas can be borrowed for local model aggregation.

Based on transfering knowledge: Hamm *et al.* [8] and Papernot *et al.* [13,14] use majority voting from local models to label auxiliary public data, then use those labelled public data to train a secure model. However, there are limitations, firstly, public data with the same distribution as training datasets is not always available, especially when involving sensitive data. In most cases, public data has different distribution with the training data.

Based on differential privacy: Some schemes choose to release local models' parameters to aggregate. To prevent private information leaking from local parameters, some works [1,18] use differential privacy by adding noise to local models' parameters. With differential privacy, those schemes achieve differentially private machine learning models. However, those schemes can only be applied on local models with exactly the same type and structure.

6 Conclusion

Motivated by securely aggregating local models with different types, we design an aggregation scheme that allows different types of local models train on their datasets privately. We propose differentially private GAN and transfer knowledge from local models to the global model. We use differentially private GANs to let local parties generate synthetic samples and all local models predict on the generated samples, then we use the majority voting count as the label. By combining generating differentially private synthetic data and querying local models' predictions, we transfer knowledge from local models to the global model. With good performance, our scheme is accurate, efficient and practical, we believe our scheme can be widely applied in the very near future.

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 308– 318. ACM (2016)
[2] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191. ACM (2017)
[3] Dwork, C., Lei, J.: Differential privacy and robust statistics. In: STOC. vol. 9, pp. 371–380 (2009)
[4] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. pp. 265–284. Springer (2006)
[5] Dwork, C., Rothblum, G.N.: Concentrated differential privacy. arXiv preprint arXiv:1603.01887 (2016)
[6] Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 51–60. IEEE (2010)
[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
[8] Hamm, J., Cao, Y., Belkin, M.: Learning privately from multiparty data. In: International Conference on Machine Learning. pp. 555–563 (2016)
[9] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
[10] Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: Kdd. vol. 96, pp. 202–207 (1996)
[11] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
[12] Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 19–38. IEEE (2017)
[13] Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., Talwar, K.: Semisupervised knowledge transfer for deep learning from private training data. arXiv preprint arXiv:1610.05755 (2016)
[14] Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Erlingsson, U.:´ Scalable private learning with pate. arXiv preprint arXiv:1802.08908 (2018)
[15] Pathak, M., Rane, S., Raj, B.: Multiparty differential privacy via aggregation of locally trained classifiers. In: Advances in Neural Information Processing Systems. pp. 1876–1884 (2010)
[16] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434
[17] (2015)
[18] Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. arXiv preprint arXiv:1401.4082 (2014)
[19] Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 1310–1321. ACM (2015)
[20] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)

A Appendix

Theorem 3. *Suppose that $f : \mathcal{D} \to \mathbb{R}^p$, with $\|f(\cdot)\|_2 \le C$, $C = 1$. Let $\sigma \ge 1$ and let $\mathcal{J}$ be a sample from $[n]$, where each $i \in [n]$ is chosen independently with probability $q < \frac{1}{16\sigma}$. Then for any positive integer $\lambda \le \sigma^2 \ln \frac{1}{q\sigma}$, the mechanism $\mathcal{M}_t(d) = \sum_{i \in J} f(d_i) + \mathcal{N}(0, C^2\sigma^2)$ satisfies*

$$\alpha_{\mathcal{M}_t}(\lambda) \le \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3) \tag{19}$$

Proof. Fix d' and let $d = d' \cup \{d_n\}$. Without loss of generality, $f(d_n) = \boldsymbol{e_1}$ and $\sum_{i \in J \setminus [n]} f(d_i) = \mathbf{0}$. Thus $\mathcal{M}_t(d)$ and $\mathcal{M}_t(d')$ are distributed identically except for the first coordinate. Therefore, we have a one-dimensional problem. Let μ_0 denote the probability density function of $\mathcal{N}(0, \sigma^2)$ and let μ_1 denote the probability density function of $\mathcal{N}(1, \sigma^2)$. Thus:

$$\begin{aligned} \mathcal{M}_t(d') &\sim \mu_0 \\ \mathcal{M}_t(d) &\sim \mu \triangleq (1-q)\mu_0 + q\mu_1 \end{aligned} \tag{20}$$

We need these two inequality

$$\mathbb{E}_{z\sim\mu}[(\mu(z)/\mu_0(z))^\lambda] \le \alpha \tag{21}$$

$$\mathbb{E}_{z\sim\mu_0}[(\mu_0(z)/\mu(z))^\lambda] \le \alpha \tag{22}$$

the parameter α will be determined later.

We will use the same method to prove those two inequality above. Assume we have two distributions ν_0 and ν_1, and we need to bound

$$\mathbb{E}_{z\sim\nu_0}[(\nu_0(z)/\nu_1(z))^\lambda] = \mathbb{E}_{z\sim\nu_1}[(\nu_0(z)/\nu_1(z))^{\lambda+1}] \tag{23}$$

By using binomial expansion, we have

$$\begin{aligned} \mathbb{E}_{z\sim\nu_1}[(\nu_0(z)/\nu_1(z))^{\lambda+1}] &= \mathbb{E}_{z\sim\nu_1}[(1 + (\nu_0(z) - \nu_1(z))/\nu_1(z))^{\lambda+1}] \\ &= \sum_{t=0}^{\lambda+1} \binom{\lambda+1}{t} \mathbb{E}_{z\sim\nu_1}[((\nu_0(z) - \nu_1(z))/\nu_1(z))^{\lambda+1}] \end{aligned} \tag{24}$$

$$\begin{aligned}
\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] &= q^2\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu(z)}\right)^2\right] \\
&= q^2\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu(z)}dz \\
&\leq \frac{q^2}{1-q}\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu_0(z)}dz \\
&= \frac{q^2}{1-q}\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right]
\end{aligned} \tag{26}$$

As the fact that for any $a \in \mathbb{R}$, $\mathbb{E}_{z\sim\mu_0}\exp(2az/2\sigma^2) = \exp(a^2/2\sigma^2)$. We have

$$\begin{aligned}
&\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right] = \mathbb{E}_{z\sim\mu_0}\left[\left(1-\exp(\frac{2z-1}{2\sigma^2})\right)^2\right] \\
&= 1-2\mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{2z-1}{2\sigma^2})\right]+\mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{4z-2}{2\sigma^2})\right] \\
&= 1-2\exp(\frac{1}{2\sigma^2})\cdot\exp(\frac{-1}{2\sigma^2})+\exp(\frac{4}{2\sigma^2})\cdot\exp(\frac{-2}{2\sigma^2}) \\
&= \exp(\frac{1}{\sigma^2})-1
\end{aligned} \tag{27}$$

Therefore, the third term in the binomial expansion (Equation (24))

$$\binom{1+\lambda}{2}\mathbb{E}_{z\in\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] \leq \frac{\lambda(\lambda+1)q^2}{(1-q)\sigma^2} \tag{28}$$

The first term in Equation (24) is 1, and the second term is

$$\begin{aligned}
\mathbb{E}_{z\sim\nu_1}\left[\frac{\nu_0(z)-\nu_1(z)}{\nu_1(z)}\right] = \int_{-\infty}^{\infty}\nu_1(z)\frac{\nu_0(z)-\nu_1(z)}{\nu_1(z)}dz &= \int_{-\infty}^{\infty}\nu_0(z)dz - \int_{-\infty}^{\infty}\nu_1(z)dz \\
&= 1-1=0
\end{aligned} \tag{25}$$

To prove Theorem 3, it can be seen that for both $\nu_0 = \mu$, $\nu_1 = \mu_0$ and $\nu_0 = \mu_0$, $\nu_1 = \mu$, the third term is bounded by $q^2\lambda(\lambda+1)/(1-q)\sigma^2$ and dominating the sum of the remaining terms. We take ($\nu_0 = \mu_0$, $\nu_1 = \mu$) as an example to prove, similarly, the other case can be proved in the same way.

To compute the upper bound for the third term in Equation (24), we note that $\mu(z) \geq (1-q)\mu_0(z)$, and we have

We use standard calculus to bound the remaining terms:

$$\forall z \leq 0 : |\mu_0(z) - \mu_1(z)| \leq -(z-1)\mu_0(z)/\sigma^2 \tag{29}$$

$$\forall z \geq 1 : |\mu_0(z) - \mu_1(z)| \leq z\mu_1(z)/\sigma^2 \tag{30}$$

$$\forall 0 \leq z \leq 1 : |\mu_0(z) - \mu_1(z)| \leq \mu_0(z)(\exp(1/2\sigma^2) - 1) \leq \mu_0(z)/\sigma^2 \tag{31}$$

Then we can get

$$\begin{aligned}\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right] \leq & \int_{-\infty}^{0}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz \\ & + \int_{0}^{1}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz \\ & + \int_{1}^{\infty}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz\end{aligned} \tag{32}$$

We consider these terms individually. By using these three observations: (1) $\mu_0 - \mu = q(\mu_0 - \mu_1)$, (2)$\mu \geq (1-q)\mu_0$ and (3) $\mathbb{E}_{\mu_0}[|z|^t] \leq \sigma^t(t-1)!!$. As we can see, the first term can be bounded by

$$\frac{q^t}{(1-q)^{t-1}\sigma^{2t}}\int_{-\infty}^{0}\mu_0(z)|z-1|^t dz \leq \frac{(2q)^t(t-1)!!}{2(1-q)^{t-1}\sigma^t} \tag{33}$$

$$\begin{aligned}\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] &= q^2\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu(z)}\right)^2\right] \\ &= q^2\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu(z)}dz \\ &\leq \frac{q^2}{1-q}\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu_0(z)}dz \\ &= \frac{q^2}{1-q}\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right]\end{aligned} \tag{26}$$

As the fact that for any $a \in \mathbb{R}$, $\mathbb{E}_{z\sim\mu_0}\exp(2az/2\sigma^2) = \exp(a^2/2\sigma^2)$. We have

$$\begin{aligned}&\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right] = \mathbb{E}_{z\sim\mu_0}\left[\left(1-\exp(\frac{2z-1}{2\sigma^2})\right)^2\right] \\ &= 1 - 2\mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{2z-1}{2\sigma^2})\right] + \mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{4z-2}{2\sigma^2})\right] \\ &= 1 - 2\exp(\frac{1}{2\sigma^2})\cdot\exp(\frac{-1}{2\sigma^2}) + \exp(\frac{4}{2\sigma^2})\cdot\exp(\frac{-2}{2\sigma^2}) \\ &= \exp(\frac{1}{\sigma^2}) - 1\end{aligned} \tag{27}$$

Therefore, the third term in the binomial expansion (Equation (24))

$$\binom{1+\lambda}{2}\mathbb{E}_{z\in\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] \leq \frac{\lambda(\lambda+1)q^2}{(1-q)\sigma^2} \tag{28}$$

The upper bound for the second term is

$$\frac{q^t}{(1-q)^t}\int_0^1 \mu(z)\left|\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^t\right| dz \leq \frac{q^t}{(1-q)^t}\int_0^1 \mu(z)\frac{1}{\sigma^{2t}}dz \leq \frac{q^t}{(1-q)^t\sigma^{2t}} \tag{34}$$

The upper bound for the third term is

$$\begin{aligned} &\frac{q^t}{(1-q)^{t-1}\sigma^{2t}}\int_1^\infty \mu_0(z)\left(\frac{z\mu_1(z)}{\mu_0(z)}\right)^t dz \\ &\leq \frac{q^t}{(1-q)^{t-1}\sigma^{2t}}\int_1^\infty \mu_0(z)\exp((2tz-t)/2\sigma^2)z^t dz \\ &\leq \frac{q^t\exp((t^2-t)/2\sigma^2)}{(1-q)^{t-1}\sigma^{2t}}\int_0^\infty \mu_0(z-t)z^t dz \\ &\leq \frac{(2q)^t\exp((t^2-t)/2\sigma^2)(\sigma^t(t-1)!!+t^t)}{2(1-q)^{t-1}\sigma^{2t}} \end{aligned} \tag{35}$$

According to the definition and range of q, σ and λ, we can see that the three terms, and their sum, drop off geometrically fast in t for $t > 3$. Therefore, the binomial expansion (Equation (24)) is dominated by the $t = 3$ term, which is $\mathcal{O}(q^3\lambda^3/\sigma^3)$. Therefore, we can claim Theorem 3 is sound.