Learning Informative and Private Representations via Generative Adversarial Networks

Tsung-Yen Yang
Princeton University
ty3@princeton.edu

Christopher Brinton
Princeton University
cbrinton@princeton.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

Mung Chiang
Purdue University
chiang@purdue.edu

Andrew Lan
Princeton University
andrew.lan@princeton.edu

***Abstract*—It is of crucial importance to simultaneously protect against sensitive attributes in data while building predictive models. In this paper, we tackle the problem of learning representations from raw data that are i) informative and predictive of desirable variables, and ii) private and protect against adversaries that attempt to recover sensitive variables. We cast this problem under the generative adversarial network (GAN) framework and design three components: an encoder, an ally that predicts the desired variables, and an adversary that predicts the sensitive ones. As a use case, we apply our approach to learn representations of raw student clickstream event data captured as they watch lecture videos in massive open online courses (MOOCs). Through experiments on a real- world dataset collected from a MOOC, we demonstrate that our method can learn a low-dimensional representation of each user that i) excels at classifying whether a user will answer a quiz question correctly, and ii) prevents an adversary from recovering each user's identity. Our results indicate that our approach is effective in learning representations that are both informative and private.**

***Index Terms*—Generative adversarial network, Massive open online courses, Predictive models, Privacy**

I. Introduction

Predictive modeling has become a key application of machine learning, finding success in applications including computer vision [1], computational social science [2], biomedical sciences [3], and education research [4]. In the majority of these applications, the prediction task is to use data collected about users to forecast certain events or outcomes, *e.g.,* the next product a customer will buy, the risk level of a patient to a certain type of cancer, the grades of a student, and so forth. As predictive modeling becomes more ubiquitous, then, it is important to develop methods that can protect sensitive attributes in user data, especially in applications like clinical diagnosis and educational research where patient and student data are highly sensitive [5].

Advances in predictive modeling rely heavily on open-access datasets, the most famous perhaps being the Netflix Prize dataset [6] from which researchers have built recommendation systems for movie preferences. It has been shown, however, that despite actions taken to preserve anonymity, user identities in these datasets can still be reverse-engineered, potentially by analyzing related datasets for which identities have not been obfuscated. The seminal work [7], for example, developed a novel de-anonymization algorithm to unveil user information that had been anonymized in public micro-data, *e.g.,* the Netflix Prize dataset and clinical datasets. The work in [8] also demonstrated that one can identify anonymous users in online social networks with up to 90% accuracy using topological features of other social network graphs, *e.g.,* Twitter. Equipped even with sparse, auxiliary information, these works demonstrate that it is possible for an attacker to uncover sensitive attributes from user data.

Several efforts have been made in machine learning to develop algorithms that achieve reasonable predictive power while preserving user privacy. The most popular line of work, differential privacy (DP) [9], proposes to add random noise to raw data, with the noise level controlling the tradeoff between predictive quality and user privacy. The Laplace mechanism [10], which perturbs raw data with random Laplace noise, is the most widely used due to its simplicity and well-defined mathematical properties. But while the Laplace mechanism is effective in preserving user privacy, it has been empirically shown to both significantly reduce the utility of data for predictive modeling [11]–[16] and dramatically increase sample complexity for training models [17]–[19]. As a result, we consider whether a method for encoding the data can be designed to preserve privacy without damaging predictive power.

Important use cases of privacy-preserving predictive modeling occur in educational applications. This is especially the case in online learning settings, *e.g.,* massive open online courses (MOOCs), where a plethora of measurements are captured on students as they interact with course content and on discussion forums [20].[1] In recent years, MOOCs have enjoyed wide success in providing high-quality, easily accessible learning content to tens of millions of students around the globe [21]–[27]. Several popular MOOCs have seen tens of thousands of students enrolled in single sessions, showing promise of scaling up learning at reasonable costs. Predictive modeling is important in MOOCs since i) it can help students monitor their learning progress [20], [28]–[31], and ii) it can help instructors identify which students are at risk of early dropout and act accordingly to retain them [32], [33]. The behavior and performance data needed to make these predictions, however, is sensitive, with many students preferring to remain anonymous to their peers. For MOOC service providers, then, both predictive modeling and privacy

[1] See, *e.g.,* coursera.org, edx.org.

preservation are crucial [5], [34].

A. Contributions

In this paper, we propose a novel methodology (Section II) for learning representations from raw user data that i) enables predictive modeling on desirable attributes while ii) preventing adversaries from recovering sensitive attributes. Casting the problem under the generative adversarial network (GAN) framework, we develop three model components: i) an encoder network that takes raw user data as input and generates learned representations as output, ii) an ally network that takes learned representations as input and predicts desired variables as output, and iii) an adversary network that takes representations as input and predicts sensitive variables as output. The encoder accommodates both static and time-varying data attributes.

We frame a popular MOOC predictive modeling use case in terms of our method (Section III). In this scenario, the users are students, the raw student data corresponds to their clickstream events generated as they watch lecture videos, the desired variables correspond to whether they are correct on first attempt (CFA) in answering quiz questions, and the sensitive attribute correspond to their identities. We explore two different settings for the generator network: first, we define a series of static, hand-crafted features summarizing the clickstreams as raw user data and use a corresponding fully-connected neural network as the encoder; second, we input the time-varying clickstream events directly and use a corresponding long short-term memory (LSTM) network as the encoder.

In evaluating our methodology (Section IV) on a MOOC dataset consisting of roughly 315,000 clickstream events generated from 4,000 students, we find that it can learn representations of raw user data that outperform hand-crafted features by 20% in terms of CFA prediction, while simultaneously reducing the possibility for adversaries to recover user identities by 30%. Furthermore, it outperforms the Laplace mechanism by 20% in terms of CFA prediction at the same privacy level. These results imply that our approach can be used to learn informative and private representations of raw user data that enable effective and safe predictive modeling.

Overall, our method separates itself from DP in two aspects. First, it is data-dependent, *i.e.,* it learns representations from user data that exhibit greater predictive power and better privacy. Second, it directly uses raw user data without relying on feature engineering; this can be a bottleneck for DP with certain data types, *e.g.,* data streams.

B. Related Work

A few previous works [35]–[37] have leveraged adversarial training to learn representations that prevent an adversary from uncovering sensitive information. However, these works mainly focus on obfuscating a single specific characteristic of an image, such as preventing a classifier from identifying a specific scene [35], removing QR code added in the original image [36], and blurring artificially superimposed text added in facial pictures [37]. Our method, by contrast, can safeguard sensitive information when data are used for storage or transmission.

Our work is most closely related to the work in [38], which also uses a GAN formulation to learn private representations from a published dataset. However, it formulates a minimax-constrained optimization problem by considering two model parts: a privatizer, which outputs private representations by injecting noise based on some constraints, and an adversary, which learns sensitive attributes from representations. In contrast, our framework learns representations automatically without adding noise and constraints. Moreover, we test our framework on a real-world dataset to demonstrate the feasibility of our model, which is not done in [38].

II. Learning Representation Methodology

In this section, we develop our method for learning representations of raw data that prevent adversaries from de-anonymizing sensitive attributes while preserving the predictive quality of desirable attributes. Specifically, we exploit the framework of generative adversarial networks (GAN), with our method being comprised of three modules: i) an encoder network (Section II-B) that outputs learned representations based on input of raw user data, ii) an ally network (Section II-C) that preserves the predictive quality of desirable attributes in the learned representations, and iii) an adversary network (Section II-D) that recovers information of sensitive attributes in the learned representations. Figure 1 summarizes the interconnection of these modules: the objective is for the encoder network to generate representations that minimize the loss function of the ally network while maximizing the the loss function of adversary network.

One advantage of our method is that it is reasonably model-agnostic: each module can instantiate specific differential functions (e.g., neural networks) based on the needs of the particular application. Consider an application in speech recognition, for example, where the objective is to predict the words a user stated from the captured audio signal without leaking sensitive personal information. The ally and adversary networks may use recurrent neural networks (RNN) for this, given their demonstrated superior quality in this application. As another example, researchers in healthcare interested in finding the connection between medical images and cancer without leaking sensitive patient information may employ convolutional neural networks, given their demonstrated superior quality in this application. In this paper, we express the modules in terms of neural networks due to their expressiveness and flexibility.

A. User-Item Data Pairs

Let the dataset of interest consist of U users and V items, with user u's interaction with item v forming the user-item pair (u, v) that possesses

1) a raw data attribute vector $\boldsymbol{f}_{u,v}$ of dimension K, i.e., with K attributes,
2) a binary classification label $y_{u,v} \in \{0, 1\}$, which we aim to predict, and

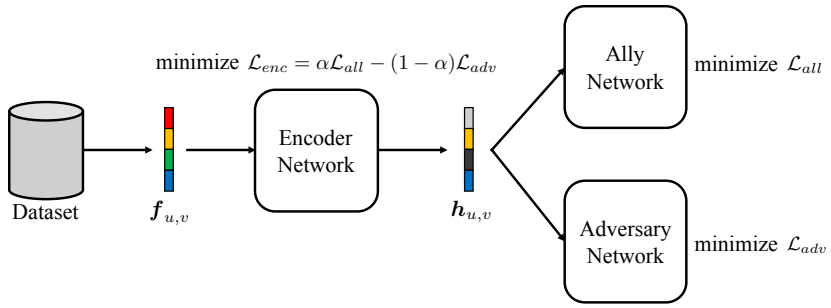


Fig. 1: The methodology we develop in this paper. An encoder network that takes raw data $\boldsymbol{f}_{u,v}$ as input and generates representations $\boldsymbol{h}_{u,v}$ to minimize $\mathcal{L}_{enc}$, an ally network that takes $\boldsymbol{h}_{u,v}$ as input and predicts desirable attributes as output, and an adversary network that takes $\boldsymbol{h}_{u,v}$ as input and predicts sensitive attributes.

3) a sensitive label $s_{u,v} \in \{1, 2, ..., S\}$, which we aim to obfuscate.

While the exact K attributes will be specific to the application, we distinguish between those that are time-varying and those that are static. When the attributes are time-varying, they will be denoted $\boldsymbol{f}_{u,v}(i)$, where $i \in \{1, ..., L_{u,v}\}$ is the (discrete) time index and $L_{u,v}$ is the length of the time series sequence for (u, v). In Section III, we will specify the determination of both static $\boldsymbol{f}_{u,v}$ and time-varying $\boldsymbol{f}_{u,v}(i)$ for our application to online education, where items are lecture videos.

B. Encoder Network

We consider a different structure of the encoder network depending on whether the attributes are static or time varying: (i) a fully-connected neural network is used when the inputs are $\boldsymbol{f}_{u,v}$, and (ii) a long short-term memory (LSTM) network is used when the inputs are $\boldsymbol{f}_{u,v}(i)$. The objective in each case is for the encoder to generate a learning representation $\boldsymbol{h}_{u,v}$ for each user-item pair from the input, raw data attributes, treating the dimension N of the representation as a model parameter.

1) Static attributes $\boldsymbol{f}_{u,v}$: In the case of static attributes, the encoder for each (u, v) is a fully connected network:

$$\begin{aligned} \boldsymbol{h}_0 =& \sigma(\boldsymbol{W}_0^T \boldsymbol{f}_{u,v} + \boldsymbol{b}_0) \\ \boldsymbol{h}_1 =& \sigma(\boldsymbol{W}_1^T \boldsymbol{h}_0 + \boldsymbol{b}_1) \\ & \vdots \\ \boldsymbol{h}_{u,v} = \boldsymbol{h}_L =& \sigma(\boldsymbol{W}_L^T \boldsymbol{h}_{L-1} + \boldsymbol{b}_L). \end{aligned} \tag{1}$$

Here, L denotes the number of layers, the matrix $\boldsymbol{W}_l$ and bias vector $\boldsymbol{b}_l$ are the parameters of the lth layer, and $\sigma(\cdot)$ can be chosen as any commonly-used nonlinearity. $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ are the same for all (u, v) pairs.

2) Time-varying attributes $\boldsymbol{f}_{u,v}(i)$: LSTMs are known for their ability to capture dependencies over long time periods [39], e.g., from time series data, motivating their use for time-varying input attributes. Specifically, our forget gate $\boldsymbol{g}$, input gate $\boldsymbol{i}$, and output gate $\boldsymbol{o}$ vectors are defined at each time step i as

$$\begin{aligned} \boldsymbol{g}_{u,v}(i) &= \sigma(\boldsymbol{W}_g \boldsymbol{g}_{u,v}(i) + \boldsymbol{U}_f \mathbf{h}_{u,v}(i-1) + \boldsymbol{b}_g), \\ \boldsymbol{i}_{u,v}(i) &= \sigma(\boldsymbol{W}_i \boldsymbol{f}_{u,v}(i) + \boldsymbol{U}_i \mathbf{h}_{u,v}(i-1) + \boldsymbol{b}_i), \\ \boldsymbol{o}_{u,v}(i) &= \varphi(\boldsymbol{W}_o \boldsymbol{f}_{u,v}(i) + \boldsymbol{U}_o (\boldsymbol{i}_{u,v}(i) \odot \mathbf{h}_{u,v}(i-1)) + \boldsymbol{b}_o). \end{aligned} \tag{2}$$

Here, $\varphi(\cdot)$ and $\sigma(\cdot)$ are the tanh and sigmoid functions, respectively. $\odot$ denotes element-wise vector multiplication and matrix multiplication. The matrices $\boldsymbol{W}_g$, $\boldsymbol{U}_f$, $\boldsymbol{W}_i$, $\boldsymbol{U}_i$, $\boldsymbol{W}_o$, and $\boldsymbol{U}_o$ as well as the vectors $\boldsymbol{b}_g, \boldsymbol{b}_i$, and $\boldsymbol{b}_o$ are model parameters. With this, $\boldsymbol{h}$ is updated in each time step as

$$\boldsymbol{h}_{u,v}(i) = \boldsymbol{g}_{u,v}(i) \odot \boldsymbol{h}_{u,v}(i-1) + (\mathbf{1} - \boldsymbol{g}_{u,v}(i-1)) \odot \boldsymbol{o}_{u,v}(i). \tag{3}$$

The forget vector $\boldsymbol{g}$ thus controls the degree to which the prior state $\boldsymbol{h}(i-1)$ and attributes $\boldsymbol{f}(0), ..., \boldsymbol{f}(i-1)$ will be used in the new state $\boldsymbol{h}(i)$, while the input gate $\boldsymbol{i}$ and output gate $\boldsymbol{o}$ together specify the degree to which the current input values $\boldsymbol{f}_{u,v}(i)$ will propagate to the update of the new state. The final state $\boldsymbol{h}_{u,v}(L_{u,v})$ at the last time index $L_{u,v}$ is taken as the learning representation for (u, v).

Encoding all the information in a long sequence into a single final state $\boldsymbol{h}_{u,v}(L_{u,v})$, however, might be unrealistic. To investigate this, we exploit the idea of an attention mechanism introduced by [40]. Instead of forcing the network to encode all the information into the final state, an attention module takes all the $\boldsymbol{h}_{u,v}(i)$ as inputs and generates the learned representation $\boldsymbol{h}_{u,v}$ as outputs. As result, the model is capable of attending to different parts of the sequence. Formally, an attention module is defined as

$$\begin{aligned} e_{u,v}(i) &= \boldsymbol{a}_{u,v}(i)^T \varphi(\boldsymbol{W}_a \boldsymbol{h}_{u,v}(i)) \\ \gamma_{u,v}(i) &= \frac{\exp(e_{u,v}(i))}{\sum_j \exp(e_{u,v}(j))} \\ \boldsymbol{h}_{u,v} &= \sum_i \gamma_{u,v}(i) \boldsymbol{h}_{u,v}(i) \end{aligned} \tag{4}$$

where $\sum_i \gamma_{u,v}(i) = 1$ is the weight of $\boldsymbol{h}_{u,v}(i)$. We will compare the performance of these two methods for generating representations from time-varying attributes in Section IV.

Loss function. For model parameter training, we define the loss function of the encoder ($\mathcal{L}_{\text{enc}}$) in terms of the losses of the ally ($\mathcal{L}_{\text{all}}$) and adversary ($\mathcal{L}_{\text{adv}}$) networks as

$$\mathcal{L}_{\text{enc}} = \alpha \mathcal{L}_{\text{all}} - (1-\alpha) \mathcal{L}_{\text{adv}}, \tag{5}$$

where $\alpha \in [0, 1]$ is a tradeoff parameter controlling the weight of the ally versus the adversary. An objective of minimizing $\mathcal{L}_{\text{enc}}$ will thus guide the encoder to minimize $\mathcal{L}_{\text{all}}$ of the ally while maximizing $\mathcal{L}_{\text{adv}}$ of the adversary. $\mathcal{L}_{\text{all}}$ and $\mathcal{L}_{\text{adv}}$ will be specified in Sections II-C and II-D.

C. Ally Network

The ally network functions as a predictor with mapping $\mathcal{F} : \boldsymbol{h}_{u,v} \rightarrow y_{u,v}$, i.e., predicting the class $y_{u,v}$ of (u, v) based on the learned representation $\boldsymbol{h}_{u,v}$. With a fully-connected L-layer neural network, the ally is specified as

$$\begin{aligned} \boldsymbol{z}_0 =& \sigma(\boldsymbol{W}_0^T \boldsymbol{h}_{u,v} + \boldsymbol{b}_0) \\ \boldsymbol{z}_1 =& \sigma(\boldsymbol{W}_1^T \boldsymbol{z}_0 + \boldsymbol{b}_1) \\ & \vdots \\ y'_{u,v} =& \sigma(\boldsymbol{w}_L^T \boldsymbol{z}_{L-1} + b_L) \end{aligned} \tag{6}$$

where $y'_{u,v} \in [0, 1]$ is the probability prediction of $y_{u,v}$. The matrices $\boldsymbol{W}_0, ..., \boldsymbol{W}_{L-1}$, vectors $\boldsymbol{w}_L, \boldsymbol{z}_0, ..., \boldsymbol{z}_{L-1}$ and scalar b_L are model parameters. From this, we define the loss function of the ally network over any given set Ω of (u, v) pairs in the training set as

$$\mathcal{L}_{\text{all}} = -\sum_{(u,v)\in\Omega} \left(y_{u,v} \ln(\sigma(y'_{u,v})) + (1 - y_{u,v}) \ln(1 - \sigma(y'_{u,v})) \right), \tag{7}$$

i.e., the sigmoid cross entropy between the predictions and targets over Ω. The choice of Ω will be discussed with model training in Section IV-A.

D. Adversary Network

The adversary network also functions as a predictor, attempting to predict the sensitive information of the (u, v) pair from the learned representation $\boldsymbol{h}_{u,v}$, i.e., $\mathcal{F} : \boldsymbol{h}_{u,v} \rightarrow s_{u,v}$. Similar to the ally, we design the adversary as a fully connected network:

$$\begin{aligned} \boldsymbol{z}_0 =& \sigma(\boldsymbol{W}_0^T \boldsymbol{h}_{u,v} + \boldsymbol{b}_0) \\ \boldsymbol{z}_1 =& \sigma(\boldsymbol{W}_1^T \boldsymbol{z}_0 + \boldsymbol{b}_1) \\ & \vdots \\ \boldsymbol{s}'_{u,v} =& softmax(\boldsymbol{W}_L^T \boldsymbol{z}_{L-1} + \boldsymbol{b}_L) \end{aligned} \tag{8}$$

where $\boldsymbol{s}'_{u,v} \in [0, 1]^S$ is the vector of predicted probabilities for $s_{u,v}$, i.e., the jth element $s'_{u,v,j}$ is the predicted probability that $s_{u,v} = j$, with $\mathbf{1}^T \boldsymbol{s}'_{u,v} = 1$. The loss function is then the sigmoid cross entropy over the classes:

$$\mathcal{L}_{\text{adv}} = -\sum_{(u,v)\in\Omega} \sum_j s_{u,v,j} \ln(\sigma(s'_{u,v,j})), \tag{9}$$

where we have slightly abused notation in using $\boldsymbol{s}_{u,v} = [s_{u,v,j}]$ as a one-hot encoding of $s_{u,v}$, i.e., $s_{u,v,j} = 1$ when $s_{u,v} = j$.

E. Training Algorithm

Formally, training starts with using a batch of training examples to minimize the encoder network loss function $\mathcal{L}_{\text{enc}}$, which is the linear combination of ally network loss function $\mathcal{L}_{all}$ and adversary network loss function $\mathcal{L}_{adv}$. After training an encoder network with the standard neural network backpropagation algorithm for several iterations, we fix representations generated by an encoder network and use them to minimize $\mathcal{L}_{\text{all}}$ and $\mathcal{L}_{\text{adv}}$. The structure of the loss function depends on the prediction tasks. For this paper, the loss functions $\mathcal{L}_{\text{all}}$ and $\mathcal{L}_{\text{adv}}$ are defined as sigmoid cross entropy.

Algorithm 1 GAN parameter tuning and learning representation algorithm.

Initialize the parameters of a encoder θ_{enc}, ally θ_{all} and adversary θ_{adv} network.
for each epoch **do**
  Sample a batch of training examples, $\{(f_{u_i,v_i}, h_{u_i,v_i}, y_{u_i,v_i}, u_i)\}_{i=1}^n$.
  for epoch t **do**
    for $(f_{u_i,v_i}, h_{u_i,v_i})$ in the batch **do**
      Update θ_{enc} according to $\nabla_{\theta_{\text{enc}}} \mathcal{L}_{\text{enc}}$ using optimization algorithms.
  Obtain representations h_{u_i,v_i} by θ_{enc} to form a batch of training examples, $\{(h_{u_i,v_i}, y_{u_i,v_i}, u_i)\}_{i=1}^n$.
  for epoch t **do**
    for $(h_{u_i,v_i}, y_{u_i,v_i}, u_i)$ in the batch **do**
      Update θ_{all} according to $\nabla_{\theta_{\text{all}}} \mathcal{L}_{\text{all}}$ with an input h_{u_i,v_i} and a target y_{u_i,v_i} using optimization algorithms.
      Update θ_{adv} according to $\nabla_{\theta_{\text{adv}}} \mathcal{L}_{\text{adv}}$ with an input h_{u_i,v_i} and a target u_i using optimization algorithms.
return Optimal θ^*_{enc}.

Algorithm 1 shows the full learning framework we develop in the paper. We utilize the inner loop to train the encoder, ally, and adversary networks. During the inner loop of the encoder network, we apply minibatch updates, to the network. After performing this step, the ally and adversary networks update their parameters according to representations encoded by an encoder network. This technique allows the network to stabilize during updating the parameters [41].

III. Dataset from Online Education

We now formulate our online education scenario in the context of the data framework from Section II. In doing so, we introduce the dataset that will be used for evaluation in Section IV.

A. MOOC Scenario and Data

Massive open online courses (MOOCs) typically contain sequences of lecture videos interspersed with in-video quizzes. A problem that has received recent research attention is how to use data captured on students as they watch these videos to predict their performance on the corresponding quizzes, to e.g., generate analytics for instructors on struggling students or confusing content [20], [31], [42]. Educational data, however, is considered to be rather sensitive: many students in online courses choose to be completely anonymous to their peers. It is therefore important that, while maximizing prediction quality, care be taken to obfuscate user identities in MOOC data.

Denote the sequence of questions in a MOOC as $q_1, q_2, ...$ We enforce a 1:1 correspondence between video content and

quizzes by considering all video content appearing between q_{n-1} and q_n as the (single) video v_n for question q_n. Two types of data can be collected about a student on v_n and q_n:

1) Video-watching clickstreams: While a student watches a video on a MOOC, actions available in the scrub bar include pausing, playing, changing the playback speed, and skipping to another place in the video. The resulting video-watching behavior is typically recorded as a sequence of clickstream events. Each clickstream event E_i contains the ID of the user $u(E_i)$, the identifier of the video $v(E_i)$, the type of action $e(E_i)$, the current position $p(E_i)$ of the video player, the position of the video player $p'(E_i)$ immediately before the action, the UNIX timestamp (in seconds) of occurrence $x(E_i)$, the binary state $s(E_i)$ – either playing or paused – of the video player, and the playback rate $r(E_i)$. In Section III-B we will explain how the E_i define attributes $\boldsymbol{f}_{u,v}$ in our model.

2) Question submissions: When a student submits an answer to an in-video question, the following is recorded as an event A_j: the user ID $u(A_j)$, video ID $v(A_j)$, answer submitted $a(A_j)$, timestamp $x(A_j)$, points rewarded $o(A_j)$, and maximum possible points $o_{max}(A_j)$. Measures of performance on quiz questions typically consider the student's first attempt on the question only [20], [42].

B. User-Video Data Pairs

The prediction objective is to define a mapping from student u's interaction on video v_n to their performance on question q_n. The privacy objective, on the other hand, is to prevent the ability of tying a student's interaction to their identity u. Taking items to be videos, then, we specify the three items for each user-video pair from Section II-A as follows:

1) Attribute vectors: We define two attribute vectors, one time-varying and one static.

Time-varying $\boldsymbol{f}_{u,v}(i)$. We are interested in all events prior to the first time student u answered the question for video v. This is the sequence of events $\boldsymbol{e}_{u,v} = [E_i | u(E_i) = u, v(E_i) = v, x(E_i) < X_{u,v}]$, where $X_{u,v} = \min\{x(A_j) | u(A_j) = u, v(A_j) = v\}$ is the timestamp of the first submission. The time-varying attributes $\boldsymbol{f}_{u,v}(i)$ are vectors of the sequence of event parameters: $\boldsymbol{f}_{u,v}(i) = [e(E_i), p(E_i), p'(E_i), x(E_i), s(E_i), r(E_i)]$, where $E_i = \boldsymbol{e}_{u,v}(i)$ is the ith event made by student u on video v, and the space of four actions $e(E_i) \in \{0, 1, ..., 3\}$ and two states $s(E_i) \in \{0, 1\}$ are mapped to integer sets.

Static $\boldsymbol{f}_{u,v}$. We also use the sequence $\boldsymbol{e}_{u,v}$ to define a set of nine quantities summarizing student u's behavior on video v. Following the method outlined in [42], they are as follows, with the range in each case indicated in parentheses:

(i) Fraction completed ($0 - 1$): The percentage of the video that the student played, not counting repeated intervals more than once.

(ii) Fraction of time spent (≥ 0): The amount of (real) time the student spent on the video, while playing or paused, divided by the total playback time of the video.[2]

(iii) Fraction of time played (≥ 0): The amount of the video that the student played, including repetitions, divided by its total playback time.

(iv) Fraction of time paused (≥ 0): The amount of time the student spent paused on the video, divided by its total playback time.

(v) Number of pauses (integer ≥ 0): The number of times the user paused the video.

(vi) Number of skips backward (integer ≥ 0): The number of times the user skipped backward in the video.

(vii) Number of skips forward (integer ≥ 0): The number of times the user skipped forward in the video.

(viii) Average playback rate ($0.5 - 2.0$): The time-average of the playback rates selected by the user while in the playing state.[3]

(ix) Standard deviation of playback rate ($0 - 0.75$): The standard deviation of the playback rates selected over time.

The static $\boldsymbol{f}_{u,v}$ is then a vector of these nine quantities. In the evaluation Section IV, we will compare the resulting model quality between these two types of attributes.

2) Classification labels: We quantify a student's performance on a question as whether the student was correct on their first attempt (CFA) at answering it or not (non-CFA) [20], [42]. Formally, $y_{u,v} = 1$ if $o(A_{u,v}) = o_{max}(A_{u,v})$ and $y_{u,v} = 0$ otherwise, where $A_{u,v}$ is student u's first submission on the question for video v, i.e., at timestamp $X_{u,v}$.

3) Sensitive labels: As stated, the privacy objective is to obfuscate student identities. Therefore, we set $s_{u,v} = u$.

C. Dataset and Groupings

The dataset used in this paper is from the fall 2012 offering of the course *Networks: Friends, Money, and Bytes* on the Coursera MOOC platform. This course has 92 in-video quiz questions among 20 lectures, with each lecture containing 4-5 videos. A total of 314,632 clickstream events were generated from 3,976 unique students that answered at least one question.

MOOCs are notorious for low completion rates, with many students viewing only a small portion of lectures. On average, each student watched roughly 7 of the 92 videos in this course (standard deviation $= 12.5$), with 75% answering less than 8 questions. When dividing our dataset into different groups of student-video pairs for training and evaluation, then, we need to ensure that sufficient samples are available for each student. We define different partitions of students depending on the attribute type:

Static attributes. For static $\boldsymbol{f}_{u,v}$, let $\mathcal{U}_S^{a_0} = \{u : N_u \geq a_0\}$ be the set of students u who answered at least a_0 questions, and let $\Omega_S^{a_0} = \{(u, v) : u \in \mathcal{U}_S^{a_0}\}$ be the corresponding set of student-video pairs. We consider Ω_S^2, the subset of 2,835 students that have at least 2 (u, v) pairs, and Ω_S^{45}, the 104 students with at least 45 pairs, to assess the impact of student activity level on model quality.

Time-varying attributes. For time-varying $\boldsymbol{f}_{u,v}(i)$, the individual clickstream events are samples. Letting $\Omega_T^{a_0,c_0,c_1} =$

[2]The playback time of a video is the time it takes to play through it at the default speed, *e.g.,* a 3:30 video has a playback time of 210 seconds.

[3]The video player in our dataset allowed rates between 0.75x and 2.0x the default speed.

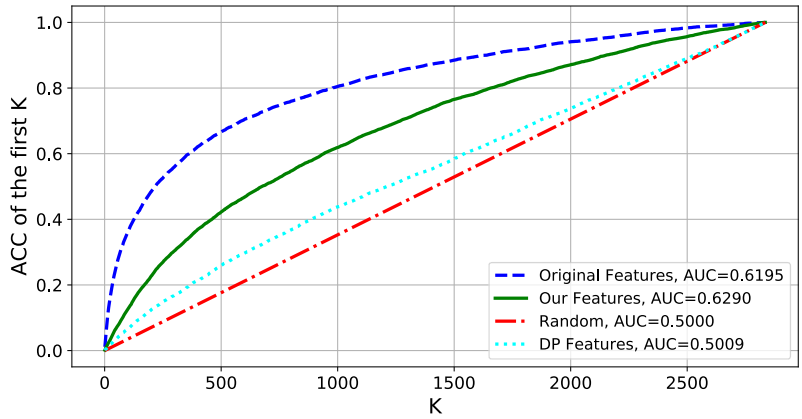


Fig. 2: Results of Top K Ranks of de-anonymization attack in Ω_S^2. The y-axis is the probability that the correct class is among the top K ranks, for each K (x-axis). We find that our representations (GAN features) outperform the original features in terms of CFA grades prediction and privacy. While the method of adding noise to features has better privacy, it suffers from low predictive quality.

$\{(u,v) : N_u \geq a_0, c_0 < L_{u,v} < c_1\}$ be the set of student-video pairs for which student u has at least a_0 questions answered and the length of the time series $L_{u,v}$ for user u on video v (i.e., the number of clickstream events in $\boldsymbol{e}_{u,v}$) is between c_0 and c_1, we set a range on the number of samples for each (u,v) pair. We consider $\Omega_T^{45,7,9}$ and $\Omega_T^{45,3,15}$ in our evaluation, to analyze the effect of larger variance in time series length on model quality; these sets correspond to 941 and 4,962 student-video pairs, respectively.

IV. Model Evaluation

We now evaluate the methodology proposed in Section II using the dataset described in Section III. After presenting our evaluation procedure (Section IV-A), we consider the following specific questions: (i) Can representations generated by our method outperform a baseline of adding noise in terms of predictive quality and privacy (Section IV-B)? (ii) How do the parameters α, of the encoder network loss function, and N, the dimension of the learned representation, impact performance (Section IV-C)? (iii) Does learning representations directly from the clickstream events provide enhancements in quality over the static quantities (Section IV-D)? We then consider analytics provided by our model on the contribution of individual clickstreams (Section IV-E).

A. Model Evaluation Setup

To evaluate our methodology, we use the following training/testing procedure, metrics, and baseline.

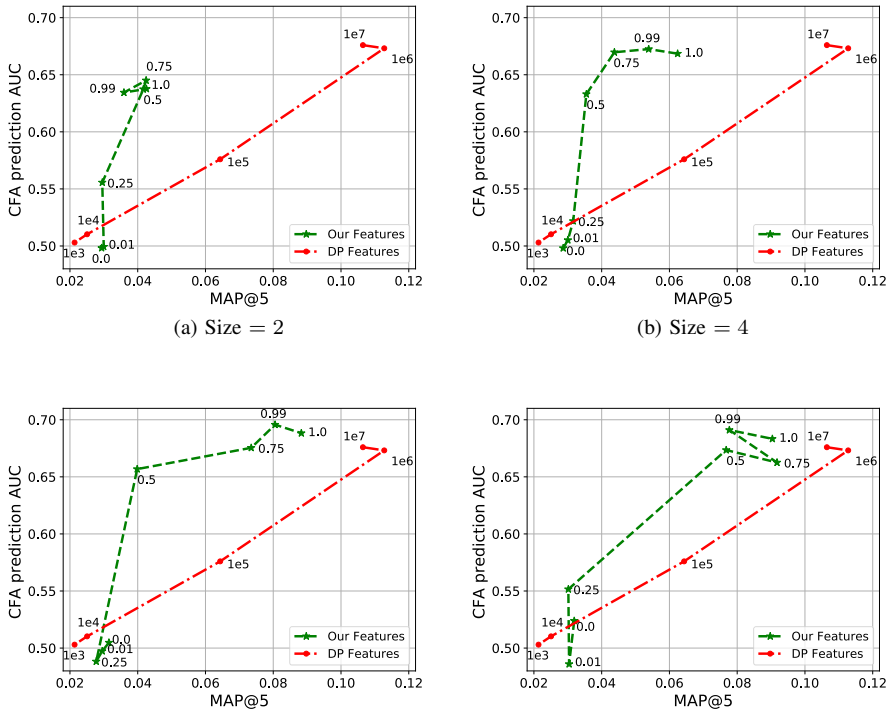
Training and testing. Following Section III, we randomly select 80% of the user-video pairs from $\Omega_S^{a_0}$ and $\Omega_T^{a_0,c_0,c_1}$ as the training set, respectively, and use the other 20% of these sets as the test set. To ensure that an adversary can have information of each individual user u to perform de-anonymization attacks, we ensure that both the training and test sets to contain user-video pairs of each student. Based on this setting, a student must answer at least two questions so that an adversary can use data in the training set to recover their identity in the test set. We perform 5-fold cross validation on each grouping of the dataset introduced in Section III-C, *i.e.,* Ω_S^2, Ω_S^{45}, $\Omega_T^{45,7,9}$, and $\Omega_T^{45,3,15}$. $\Omega_T^{45,7,9}$ has more consistent length than $\Omega_T^{45,3,15}$, which helps us to examine how the model performs under different event lengths.

On each fold, we train our model on the training set and then generate representations $\boldsymbol{h}_{u,v}$ for samples from both the training and test sets. After obtaining the $\boldsymbol{h}_{u,v}$, we train another classifier on the training set using standard classification algorithms such as multi-layer perceptron to predict CFA grades $y_{u,v}$ and user identity on the test set. For an encoder network that learns from static attributes $\boldsymbol{f}_{u,v}$, we sweep the size of representations as $N \in \{2,4,9,15\}$ since we have a total of 9 clickstream quantities, as defined in Section III-B. We also sweep the tradeoff parameter as $\alpha \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$ to see how the performance fluctuates. After parameter tuning, we set all three parts of our framework be fully-connected neural networks with two hidden layers, with sizes $[20, 10]$ in the encoder network, $[20, 10]$ in the ally network, and $[100, 50]$ in the adversary network.

Metrics. To measure the predictive quality of the learned representations, we use two metrics. First, to measure the performance of CFA prediction on a set Ω, we compute the overall accuracy (ACC), or the fraction of correct predictions:

$$\frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} \mathbb{1}\{y_{u,v} = \hat{y}_{u,v}\}$$

where $\hat{y}_{u,v} \in \{0,1\}$ is the binary prediction made based on the predicted $y'_{u,v}$ and $\mathbb{1}$ is the indicator function. Second, we compute the area under the ROC curve (AUC), which assesses the tradeoff between true and false positive rates for a



(a) Size = 2 (b) Size = 4

(c) Size = 9 (d) Size = 15

Fig. 3: Plots of MAP@5 and AUC of our framework and the method of Laplace noise in Ω_S^{45}. Each point in the curve of our features represents different α while points in the curve of DP features represents different ϵ. Our features consistently outperform DP features in terms of the utility and privacy of the data.

classifier. Random guessing has an AUC of 0.5 while a perfect model has an AUC of 1.

To measure the performance of privacy preservation, we use two other metrics: Top K Ranks [43] and mean average precision at K (MAP@K). The first metric, Top K Ranks, is motivated by the observation that the adversary's goal may only be to trim down the list of possible users from a representation, rather than recovering the exact identity of a user. As a result, a classifier takes a student-video pair's representation as input and generates a sequence of possible users as output, sorted by descending likelihood. Then, Top K Ranks simply corresponds to the accuracy of the first K predictions. The second metric, MAP@K, simply corresponds to the average precision of the first K predictions. Note that these two metrics are both in $[0, 1]$.

Baseline. We include one baseline method as the benchmark for our method. As mentioned in Section I, we use the popular Laplace mechanism in DP [10], which simply adds Laplace noise to the data. Concretely, define $GS(\Omega)$, the global sensitivity of Ω, as:

$$GS(\Omega) = \max_{\boldsymbol{f}_{u_i,v_i}, \boldsymbol{f}_{u_j,v_j} \in \Omega} ||\boldsymbol{f}_{u_i,v_i} - \boldsymbol{f}_{u_j,v_j}||_1.$$

Then we can generate private representations by

$$\boldsymbol{h}_{u,v} = \boldsymbol{f}_{u,v} + Lap\left(\frac{GS(\Omega)}{\epsilon}\right)\mathbf{1} \tag{10}$$

where the Laplace distribution $Lap(a)$ has density $p(x; a) = \frac{1}{2a}\exp(-\frac{x}{a})$ and $\mathbf{1}$ is a vector of all ones. In our experiment, we choose $\epsilon \in \{1e7, 1e6, 1e5, 1e4, 1e3\}$ since the value of $GS(\Omega)$ was observed on the order of $1e4$ in our dataset.

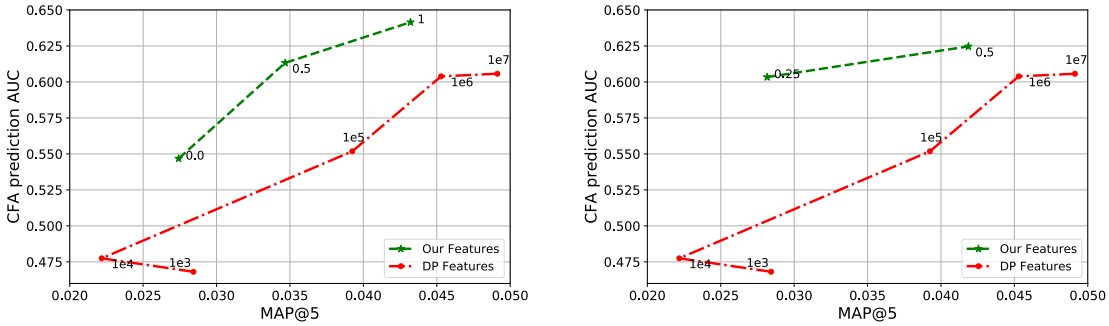
Implementation. All of the simulations are implemented using TensorFlow,[4] an open source package for neural network training and testing.

B. Overall Predictive Quality and Privacy

Figure 2 shows our most important result, which provides the full distribution over Top K ranks of Ω_S^2. Note the red dashed line indicates the baseline with random guessing and thus a straight line from 0 to 1. DP features are generated by adding Laplace noise with $\epsilon = 1e3$ to the original behavioral features. We choose $\epsilon = 1e3$ here since we observe that it corresponds to the lowest noise level that removes the utility of the data, *i.e.,* forcing AUC to approach 0.5.

Overall, the results show that our proposed method outperforms representations with Laplace noise by 20% in terms of CFA grade prediction, and even performs slightly better than the original features without adding noise. In terms of removing student identity information, it significantly outperforms the original features by reducing the probability of user identity recovery by 40%. Although it seems that

[4]https://www.tensorflow.org/



(a) Without attention

(b) With attention

Fig. 4: Plots of MAP@5 and AUC of our framework and the method of adding Laplace noise in $\Omega_T^{45,7,9}$ with and without attention module. Each point in the curve of our features represents different α while points in the curve of DP features represents different ϵ.

representations with adding Laplace noise with $\epsilon = 1e3$ have better privacy, the data utility is completely lost, with an AUC of only 0.5009. Therefore, we conclude that our framework is capable of removing sensitive information from the data without sacrificing its predictive power of desirable attributes.

C. Impact of α and N

Varying α and ϵ. Figure 3 visualizes the impact of the parameters α from the encoder loss function and ϵ from the Laplace method, with different representation sizes N. As $\alpha \to 1$, we expect an encoder network will generate representations that are more predictive of CFA grades while less effective in concealing user identities, which is consistent with the plot. For the DP baseline, larger ϵ means adding smaller amount of Laplace noise and thus leads to better predictive performance in CFA grades while lower privacy levels. Comparing these two mechanisms, we observe that our representations can achieve more than 20% improvement over DP in CFA prediction at the same privacy level when α is larger than 0.5. When α is close to 0, these two methods have similar CFA prediction quality and privacy levels. Our representations can also improve privacy by at least 30% while achieving the same CFA prediction performance, with the size of representations being 9 and α set to 0.5. These results show that our representations are more informative and private than those generated by DP, when the value of α is large enough.

Varying N. The choice of the size of representations affects several aspects of the model, including the level of compression, the amount of utility preserved, the computational resources required to encode the original features, and the storage space required for the learned representations. Figure 3 shows the results under different size of representations. The CFA prediction performance elevates up to 0.7 in terms of AUC when the size of hidden layer is larger than 9, for appropriate α values. We also find that the larger sizes make the learned representations preserve more information towards both CFA prediction and user identity with the same α value; this observation confirms the fact that the size of representations controls the amount of information contained in the data representation. In practice, then, the dimension can be tuned to obtain representations that satisfy computational and storage constraints.

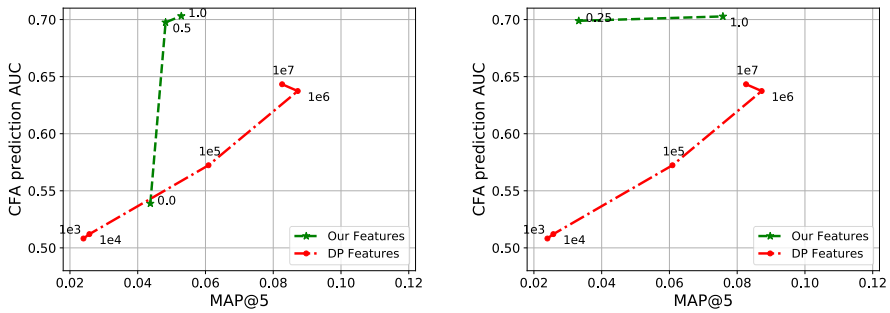
D. Static versus Time-Varying Attributes

In Sections IV-B-IV-D, we analyzed the representations generated from hand-crafted, static $\boldsymbol{f}_{u,v}$ attributes. In practice, designing such features can be a difficult and time-consuming task. Moreover, there is no guarantee that the engineered features capture all the useful information in the raw data. We therefore seek to compare model quality between these two different attributes.

To learn representations directly from the raw clickstream attributes $\boldsymbol{f}_{u,v}(i)$, we implement the LSTM network mentioned in Section II-B, which avoids feature engineering. Recall that we introduced two embedding methods to generate representations for each user; in one case, the encoder generates embeddings solely based on the hidden state corresponding to the last clickstream event, and in the other, an attention mechanism is introduced. In each case, we also concatenate the representation with another latent vector corresponding to each individual video to obtain the final representation used as inputs to the ally and adversary networks.

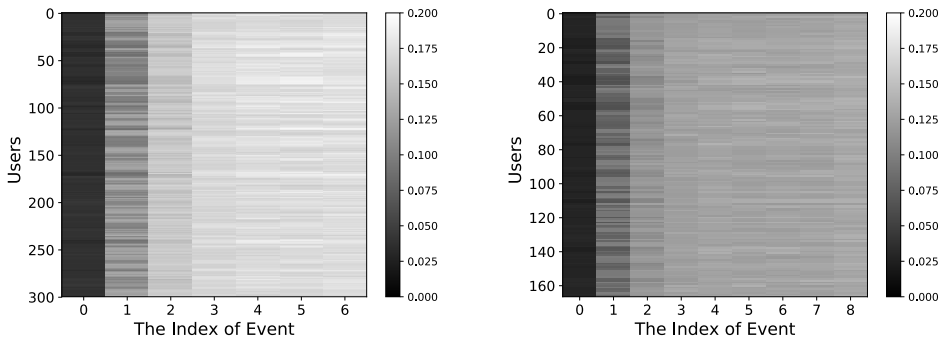
Figure 4(a) and Figure 5(a) show the results with different values of α for the non-attention mechanism and compares it to the DP baseline which adds Laplace noise to the hand-crafted clickstream features. We see that the representations learned from raw clickstreams significantly outperform the hand-crafted features in terms of the tradeoff between CFA prediction quality and user privacy.

To assess whether using only the last hidden state in the LSTM network is appropriate, especially for long clickstream sequences, we also compare performance to the attention mechanism. Figure 4(b) and Figure 5(b) show the result of the performance with an attention network on top of the LSTM network, in terms of CFA prediction quality and privacy level. We observe that the attention mechanism is capable of



(a) Without attention (b) With attention

Fig. 5: Plots of MAP@5 and AUC of our framework and the method of adding Laplace noise in $\Omega_T^{45,3,15}$ with and without attention module. Each point in the curve of our features represents different α while points in the curve of DP features represents different ϵ.



(a) Size of clickstream events = 7. (b) Size of clickstream events = 9.

Fig. 6: Examples found by $\Omega_T^{45,7,9}$. The x-axis and y-axis are the sequence of clickstream events and the samples in the dataset respectively. Each element shows the weight of the clickstream events.

generating more informative features in some cases, with up to 20% increase in CFA prediction quality on the $\Omega_T^{45,7,9}$ dataset when $\alpha = 0.25$, and with a 30% increase in privacy level when achieving the same level of CFA prediction quality at $\alpha = 0.25$ of $\Omega_T^{45,7,9}$. On the other hand, we also observe that on the $\Omega_T^{45,3,15}$ dataset, the addition of an attention module leads to no performance improvement.

E. Dynamics of Learned Representations

The attention module can give insight into which clickstream events contribute to the model quality the most. In particular, analyzing the difference between earlier events and later events in terms of their impact on the final representation can yield better understandings of student behavior. While an in-depth analysis is left for future work, in Figure 6 we visualize the weight of each clickstream event for user-video pairs with a total of 7 and 9 clickstream events. We observe smaller weights in the first and second events and larger weights in later events across these two examples. This phenomenon could be explained by the fact that most users tend to pause a video and change the play speed at the beginning of the lecture; these events are common across users and are not highly correlated with CFA and user identity. When more events are generated, the attention module focuses more on later clickstreams; this observation further confirms that using only the last hidden state as the final representation lead to quality prediction results.

V. Conclusions

In this paper, we proposed a framework for learning informative and private representations utilizing generative adversarial networks. Our framework consists of three parts: an encoder network which generates intermediate representations from raw user data, an ally network which predicts desirable attributes based on these representations, and an adversary network which predicts sensitive attributes based on these representations. Through evaluation on a real-world MOOC dataset, we have shown that our method achieves superior quality in predicting user grades from raw user clickstream events data while further reducing the de-anonymization threat, compared to differential privacy. Through our use of an attention module in the encoder network, we found that later clickstream events contribute more to a user's representation than earlier events.

Avenues of future work include i) apply our method to other applications containing sensitive data attributes, *e.g.*,

medical data, speech data, and network traffic data, ii) compare the learned representations with and without the adversary network in order to understand the key towards anonymity, and iii) experiment with cases where there are multiple desired attributes and multiple sensitive attributes.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Dec. 2012, pp. 1097–1105.

[2] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, Mar. 2007.

[3] M. Shipp, K. Ross, P. Tamayo, A. Weng, J. Kutok, R. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. Pinkus *et al.*, "Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning," *Nature Medicine*, vol. 8, no. 1, p. 68, Jan. 2002.

[4] N. Schmitt, J. Keeney, F. Oswald, T. Pleskac, A. Billington, R. Sinha, and M. Zorzie, "Prediction of 4-year college student performance using cognitive and noncognitive predictors and the impact on demographic status of admitted students." *Journal of Applied Psychology*, vol. 94, no. 6, p. 1479, Nov. 2009.

[5] J. Daries, J. Reich, J. Waldo, E. Young, J. Whittinghill, D. Seaton, A. Ho, and I. Chuang, "Privacy, anonymity, and big data in the social sciences," *Queue*, vol. 12, no. 7, p. 30, July 2014.

[6] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[7] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 111–125.

[8] ——, "De-anonymizing social networks," in *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 173–187.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory of Cryptography Conference*, Mar. 2006, pp. 265–284.

[10] ——, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[11] S. E. Fienberg, A. Rinaldo, and X. Yang, "Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables," in *International Conference on Privacy in Statistical Databases*. Springer, 2010, pp. 187–199.

[12] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1673–1693, 2012.

[13] X. He, A. Machanavajjhala, and B. Ding, "Blowfish privacy: Tuning privacy-utility trade-offs using policies," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1447–1458.

[14] A. Makhdoumi and N. Fawaz, "Privacy-utility tradeoff under statistical uncertainty," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 1627–1634.

[15] V. Rastogi, D. Suciu, and S. Hong, "The boundary between privacy and utility in data publishing," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 531–542.

[16] J. Brickell and V. Shmatikov, "The cost of privacy: destruction of data-mining utility in anonymized data publishing," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 70–78.

[17] K. Chaudhuri and D. Hsu, "Sample complexity bounds for differentially private learning," in *Proceedings of the 24th Annual Conference on Learning Theory*, 2011, pp. 155–186.

[18] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *Conference on Learning Theory*, 2012, pp. 26–1.

[19] A. Beimel, K. Nissim, and U. Stemmer, "Characterizing the sample complexity of private learners," in *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. ACM, 2013, pp. 97–110.

[20] C. Brinton and M. Chiang, "MOOC performance prediction via click-stream data and social learning networks," in *Proc. IEEE Conf. Comput. Commun.*, April 2015, pp. 2299–2307.

[21] F. G. Martin, "Will massive open online courses change how we teach?" *Comm. ACM*, vol. 55, no. 8, pp. 26–28, Aug. 2012.

[22] J. Knox, S. Bayne, H. MacLeod, J. Ross, and C. Sinclair, "MOOC pedagogy: the challenges of developing for coursera," *Online Newsletter of the Association for Learning Technologies*, Aug. 2012.

[23] R. G. Baraniuk, "Opening education," *The Bridge on Undergraduate Engineering Education*, vol. 43, no. 2, pp. 41–47, Summer 2013.

[24] J. Wilkowski, A. Deutsch, and D. Russell, "Student skill and goal achievement in the mapping with Google MOOC," in *Proc. 1st ACM Conf. on Learning at Scale*, Mar. 2014, pp. 3–10.

[25] P. Guo and K. Reinecke, "Demographic differences in how students navigate through MOOCs," in *Proc. 1st ACM Conf. on Learning at Scale*, Mar. 2014, pp. 21–30.

[26] J. Qiu, J. Tang, T. X. Liu, J. Gong, C. Zhang, Q. Zhang, and Y. Xue, "Modeling and predicting learning behavior in moocs," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, Feb. 2016, pp. 93–102.

[27] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proceedings of the 26th International Conference on World Wide Web*, Apr. 2017, pp. 765–774.

[28] X. Wang, D. Yang, M. Wen, K. Koedinger, and C. Rosé, "Investigating how student's cognitive behavior in MOOC discussion forums affect learning gains," in *Proc. Intl. Conf. Educ. Data Min.*, June 2015, pp. 226–233.

[29] S. Tomkins, A. Ramesh, and L. Getoor, "Predicting post-test performance from online student behavior: A high school MOOC case study," in *Proc. Intl. Conf. Educ. Data Min.*, June 2016, pp. 239–246.

[30] K. Koedinger, J. Kim, J. Jia, E. McLaughlin, and N. Bier, "Learning is not a spectator sport: Doing is better than watching for learning from a MOOC," in *Proc. ACM Conf. Learn at Scale*, Mar. 2015, pp. 111–120.

[31] T.-Y. Yang, C. G. Brinton, C. Joe-Wong, and M. Chiang, "Behavior-Based Grade Prediction for MOOCs Via Time Series Neural Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 5, pp. 716–728, 2017.

[32] S. Halawa, D. Greene, and J. Mitchell, "Dropout prediction in MOOCs using learner activity features," in *Proc. European MOOCs Stakeholders Summit*, Feb. 2014, pp. 58–65.

[33] J. Whitehill, J. Williams, G. Lopez, C. Coleman, and J. Reich, "Beyond prediction: Towards automatic intervention in MOOC student stop-out," in *Proc. Intl. Conf. Educ. Data Min.*, June 2015, pp. 171–178.

[34] Family educational rights and privacy act (FERPA). https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html?

[35] F. Pittaluga, S. Koppal, and A. Chakrabarti, "Learning privacy preserving encodings through adversarial training," *arXiv preprint arXiv:1802.05214*, Feb. 2018.

[36] N. Raval, A. Machanavajjhala, and L. P. Cox, "Protecting visual secrets using adversarial nets," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 1329–1332.

[37] H. Edwards and A. Storkey, "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*, 2015.

[38] C. Huang, P. Kairouzyz, X. Chen, S. L., and R. Rajagopal, "Context-aware generative adversarial privacy," *arXiv preprint arXiv:1710.09549*, Dec. 2017.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[42] A. S. Lan, C. G. Brinton, T. Yang, and M. Chiang, "Behavior-based latent variable model for learner engagement," in *Proc. Intl. Conf. Educ. Data Min.*, June 2017, pp. 64–71.

[43] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song, "On the feasibility of internet-scale author identification," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 300–314.