US 20230195541A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2023/0195541 A1**

TRUONG et al. (43) **Pub. Date: Jun. 22, 2023**

(54) **SYSTEMS AND METHODS FOR SYNTHETIC DATA GENERATION**

(71) Applicant: **Capital One Services, LLC**, McLean, VA (US)

(72) Inventors: **Anh TRUONG**, Champaign, IL (US); **Fardin ABDI TAGHI ABAD**, Champaign, IL (US); **Jeremy GOODSITT**, Champaign, IL (US); **Austin WALTERS**, Savoy, IL (US); **Mark WATSON**, Urbana, IL (US); **Vincent PHAM**, Champaign, IL (US); **Noriaki TATSUMI**, Silver Spring, MD (US); **Michael WALTERS**, Brooklyn, NY (US); **Kate KEY**, Effingham, IL (US); **Reza FARIVAR**, Champaign, IL (US); **Kenneth TAYLOR**, Champaign, IL (US)

(73) Assignee: **Capital One Services, LLC**, McLean, VA (US)

(21) Appl. No.: **18/165,725**

(22) Filed: **Feb. 7, 2023**

Related U.S. Application Data

(63) Continuation of application No. 16/151,407, filed on Oct. 4, 2018, now Pat. No. 11,615,208.

(Continued)

Publication Classification

(51) **Int. Cl.**

G06F 9/54 (2006.01)
G06N 20/00 (2006.01)

(Continued)

(52) **U.S. Cl.**

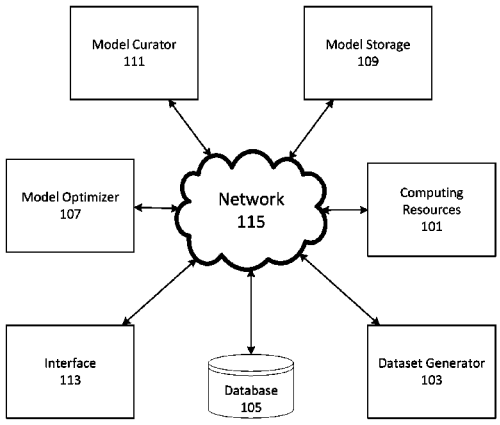
CPC ***G06F 9/541*** (2013.01); ***G06N 20/00*** (2019.01); ***G06F 17/16*** (2013.01); ***G06N 3/04*** (2013.01); ***G06F 11/3628*** (2013.01); ***G06N 3/088*** (2013.01); ***G06F 21/6254*** (2013.01); ***G06N 5/04*** (2013.01); ***G06F 17/15*** (2013.01); ***G06F 21/6245*** (2013.01); ***G06T 7/194*** (2017.01); ***G06T 7/254*** (2017.01); ***G06T 7/246*** (2017.01); ***G06T 7/248*** (2017.01); ***G06F 16/24568*** (2019.01); ***G06F 16/2237*** (2019.01); ***G06F 16/285*** (2019.01); ***G06F 16/906*** (2019.01); ***G06F 16/93*** (2019.01); ***G06F 16/90335*** (2019.01); ***G06F 16/9038*** (2019.01); ***G06F 16/90332*** (2019.01); ***G06F 16/258*** (2019.01); ***G06F 16/288*** (2019.01); ***G06F 16/283*** (2019.01); ***G06F 16/335*** (2019.01); ***G06F 16/2264*** (2019.01); ***G06F 16/2423*** (2019.01); ***G06F 16/248*** (2019.01); ***G06F 16/254*** (2019.01); ***G06F 30/20*** (2020.01); ***G06F 40/166*** (2020.01); ***G06F 40/117*** (2020.01); ***G06F 40/20*** (2020.01); ***G06F 8/71*** (2013.01); ***G06F 9/54*** (2013.01); ***G06F 9/547*** (2013.01); ***G06F 11/3608*** (2013.01); ***G06F 11/3636*** (2013.01); ***G06F 17/18*** (2013.01); ***G06F 21/552*** (2013.01); ***G06F 21/60*** (2013.01); ***G06N 7/00*** (2013.01); ***G06Q 10/04*** (2013.01); ***G06T 11/001*** (2013.01); ***H04L 63/1416*** (2013.01); ***H04L 63/1491*** (2013.01); ***H04L 67/306*** (2013.01);

(Continued)

(57) **ABSTRACT**

A cloud computing system can be configured to generate data models. A model optimizer of the cloud computing system can provision computing resources of the cloud computing system with a data model. A dataset generator of the cloud computing system can generate a synthetic dataset for training the data model. The computing resources can train the data model using the synthetic dataset. The model optimizer can store the data model and metadata of the data model in a model storage. The cloud computing system can receive production data from a data source by a production instance of the cloud computing system using a common file system. The production data can be processed using the data model by the production instance. The computing resources, the dataset generator, and the model optimizer can be hosted by separate virtual computing instances of the cloud computing system.

100



Related U.S. Application Data

(60) Provisional application No. 62/694,968, filed on Jul. 6, 2018.

Publication Classification

(51) **Int. Cl.**

G06F 17/16	(2006.01)
G06N 3/04	(2006.01)
G06F 11/36	(2006.01)
G06N 3/088	(2006.01)
G06F 21/62	(2006.01)
G06N 5/04	(2006.01)
G06F 17/15	(2006.01)
G06T 7/194	(2006.01)
G06T 7/254	(2006.01)
G06T 7/246	(2006.01)
G06F 16/2455	(2006.01)
G06F 16/22	(2006.01)
G06F 16/28	(2006.01)
G06F 16/906	(2006.01)
G06F 16/93	(2006.01)
G06F 16/903	(2006.01)
G06F 16/9038	(2006.01)
G06F 16/9032	(2006.01)
G06F 16/25	(2006.01)
G06F 16/335	(2006.01)
G06F 16/242	(2006.01)
G06F 16/248	(2006.01)
G06F 30/20	(2006.01)
G06F 40/166	(2006.01)
G06F 40/117	(2006.01)
G06F 40/20	(2006.01)
G06F 8/71	(2006.01)
G06F 17/18	(2006.01)
G06F 21/55	(2006.01)
G06F 21/60	(2006.01)
G06N 7/00	(2006.01)
G06Q 10/04	(2006.01)
G06T 11/00	(2006.01)
H04L 9/40	(2006.01)
H04L 67/306	(2006.01)
H04L 67/00	(2006.01)
H04N 21/234	(2006.01)
H04N 21/81	(2006.01)
G06N 5/00	(2006.01)
G06N 5/02	(2006.01)
G06V 30/196	(2006.01)
G06F 18/22	(2006.01)
G06F 18/23	(2006.01)
G06F 18/24	(2006.01)
G06F 18/40	(2006.01)
G06F 18/213	(2006.01)
G06F 18/214	(2006.01)
G06F 18/21	(2006.01)
G06F 18/20	(2006.01)
G06F 18/2115	(2006.01)
G06F 18/2411	(2006.01)
G06F 18/2415	(2006.01)
G06N 3/044	(2006.01)
G06N 3/045	(2006.01)
G06N 7/01	(2006.01)
G06V 30/194	(2006.01)
G06V 10/98	(2006.01)
G06V 10/70	(2006.01)
G06N 3/08	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/34*** (2013.01); ***H04N 21/23412*** (2013.01); ***H04N 21/8153*** (2013.01); ***G06N 5/00*** (2013.01); ***G06N 5/02*** (2013.01); ***G06V 30/1985*** (2022.01); ***G06F 18/22*** (2023.01); ***G06F 18/23*** (2023.01); ***G06F 18/24*** (2023.01); ***G06F 18/40*** (2023.01); ***G06F 18/213*** (2023.01); ***G06F 18/214*** (2023.01); ***G06F 18/217*** (2023.01); ***G06F 18/285*** (2023.01); ***G06F 18/2115*** (2023.01); ***G06F 18/2148*** (2023.01); ***G06F 18/2193*** (2023.01); ***G06F 18/2411*** (2023.01); ***G06F 18/2415*** (2023.01); ***G06N 3/044*** (2023.01); ***G06N 3/045*** (2023.01); ***G06N 7/01*** (2023.01); ***G06V 30/194*** (2022.01); ***G06V 10/993*** (2022.01); ***G06V 10/768*** (2022.01); ***G06N 3/08*** (2013.01); *G06T 2207/20084* (2013.01); *G06T 2207/10016* (2013.01); *G06T 2207/20081* (2013.01)

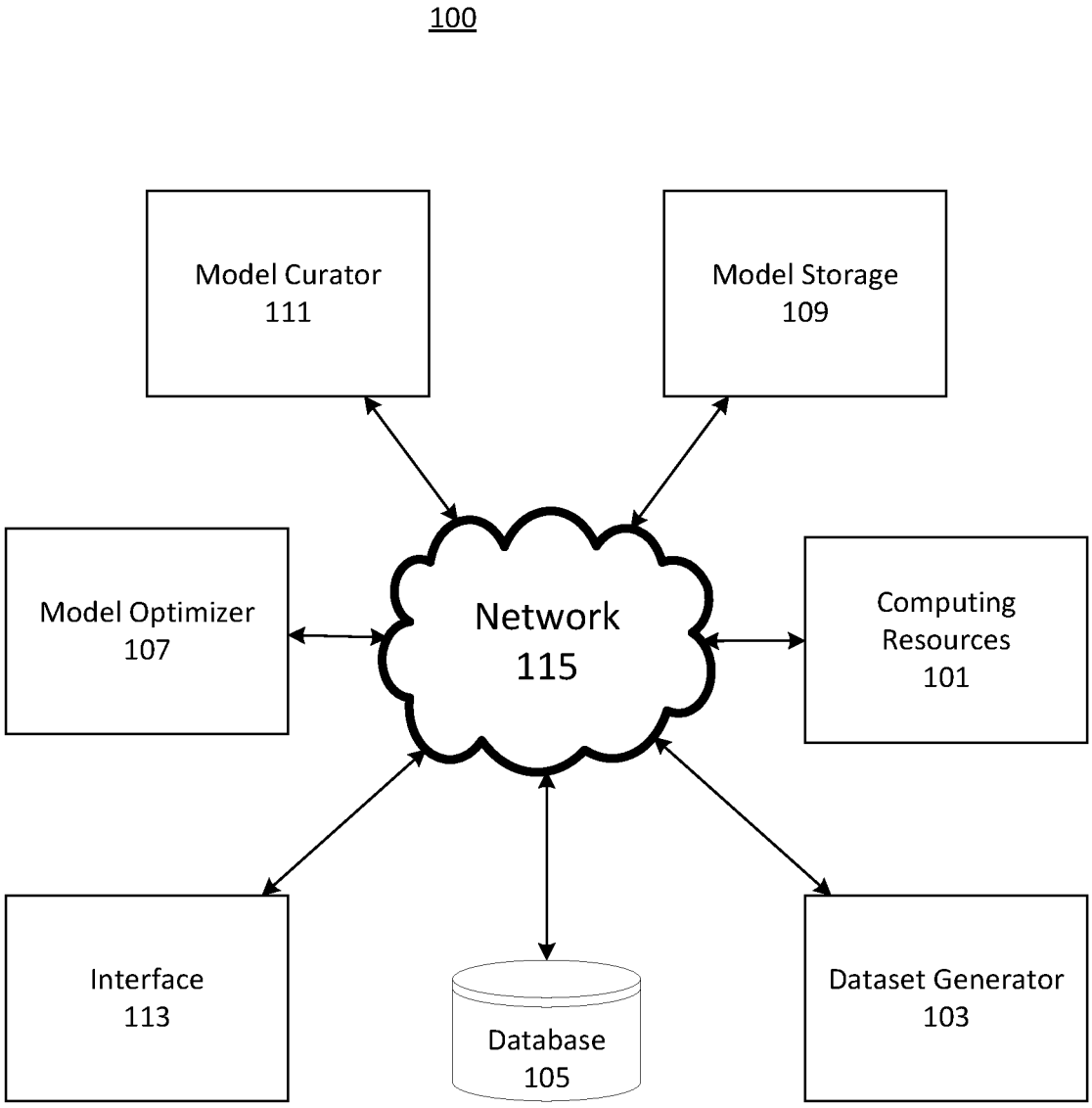

100
Model Curator
111
Model Storage
109
Model Optimizer
107
Network
115
Computing
Resources
101
Interface
113
Database
105
Dataset Generator
103

Fig. 1

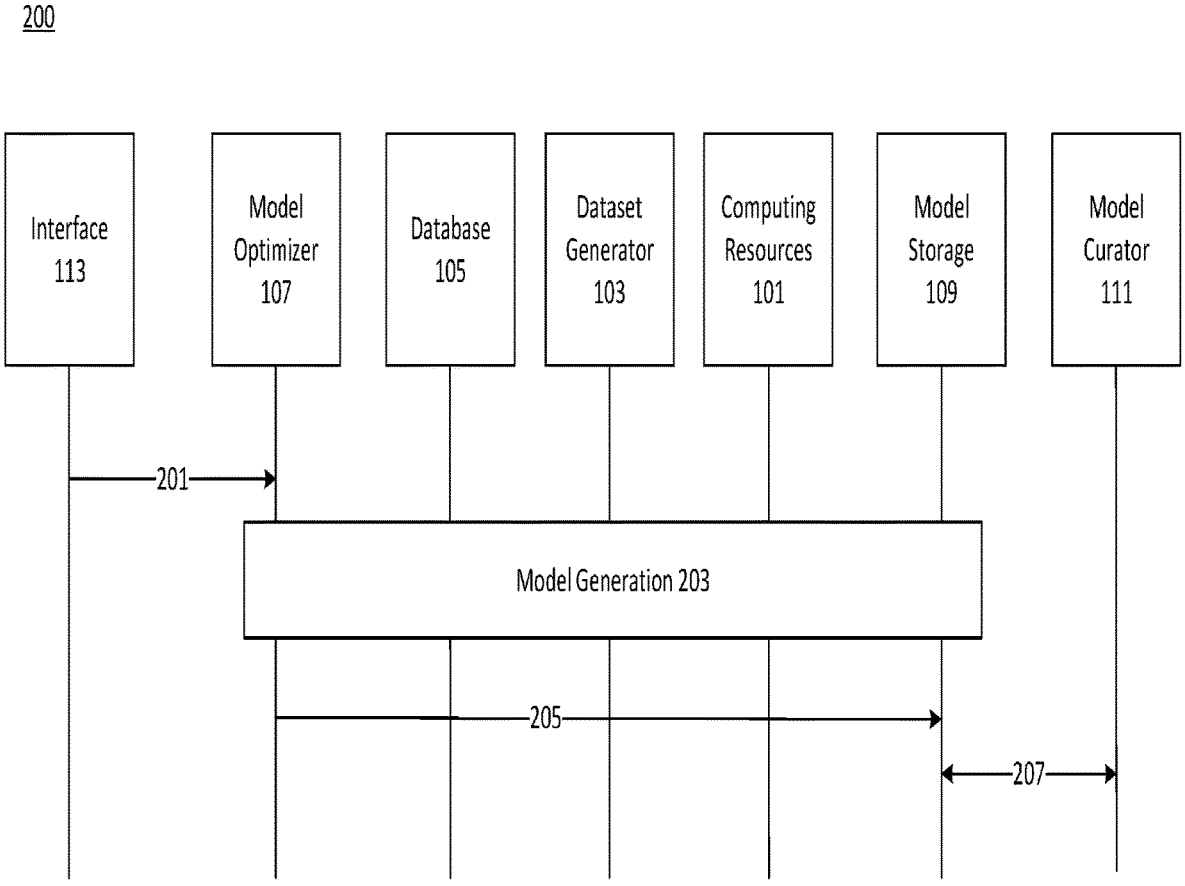

200
Interface 113
Model Optimizer 107
Database 105
Dataset Generator 103
Computing Resources 101
Model Storage 109
Model Curator 111
201
Model Generation 203
205
207

Fig. 2

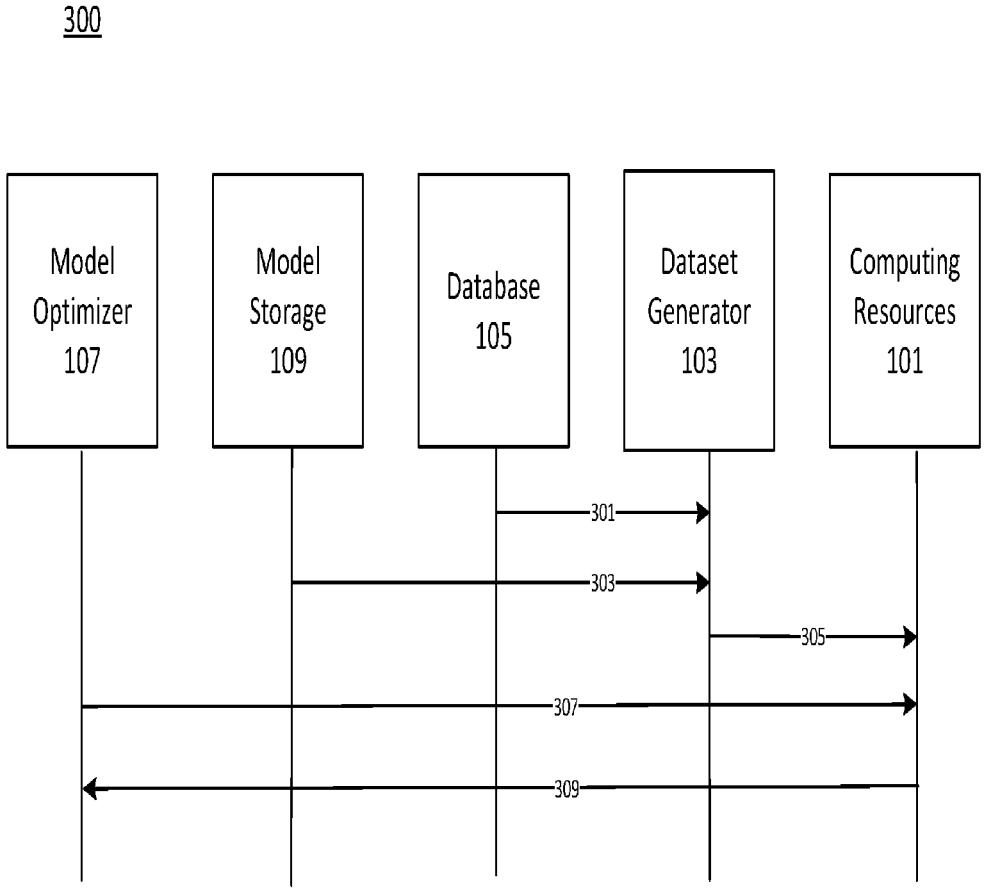

300
Model Optimizer 107
Model Storage 109
Database 105
Dataset Generator 103
Computing Resources 101
301
303
305
307
309

Fig. 3

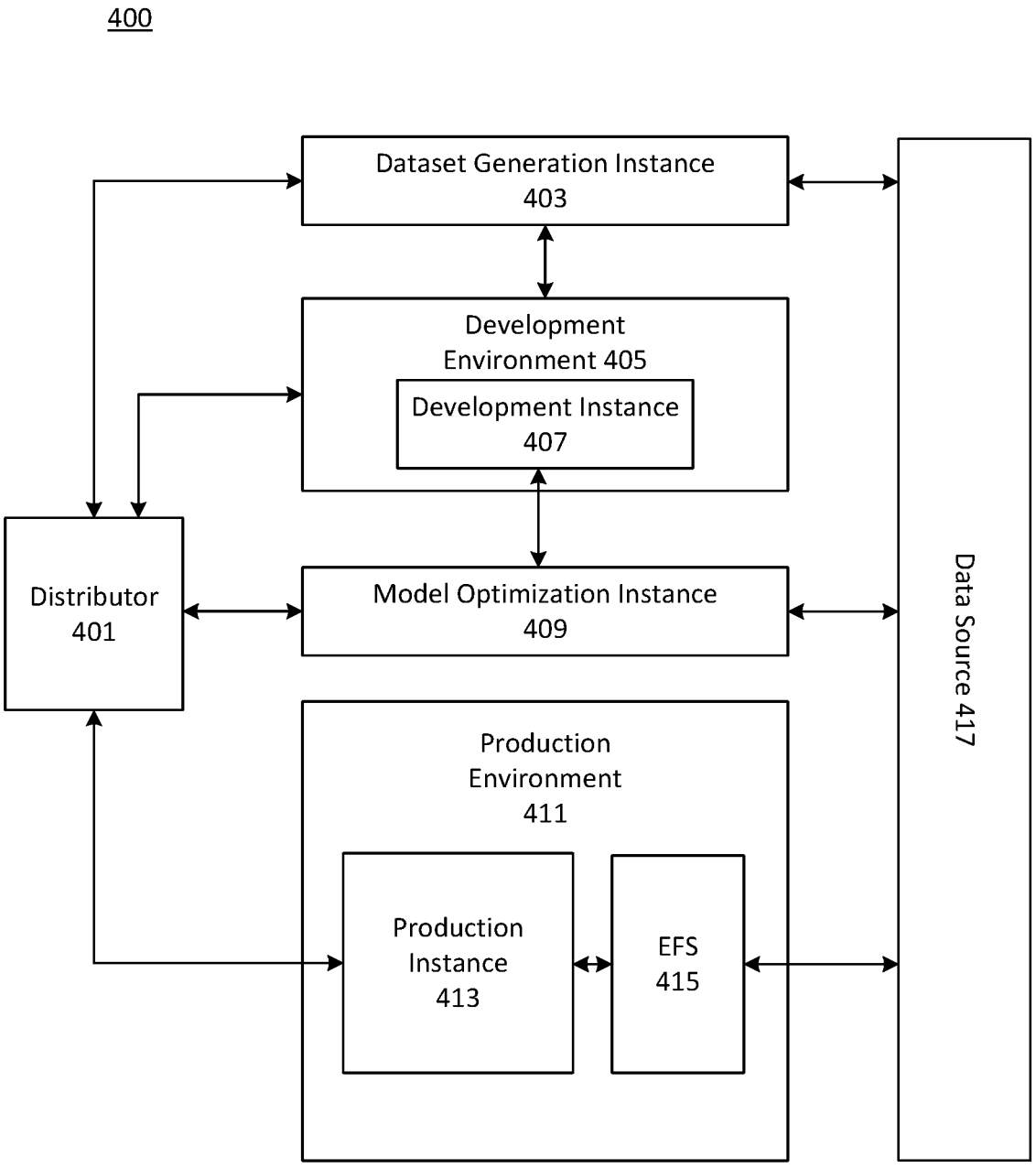

400
Dataset Generation Instance
403
Development
Environment 405
Development Instance
407
Distributor
401
Model Optimization Instance
409
Production
Environment
411
Production
Instance
413
EFS
415
Data Source 417

Fig. 4

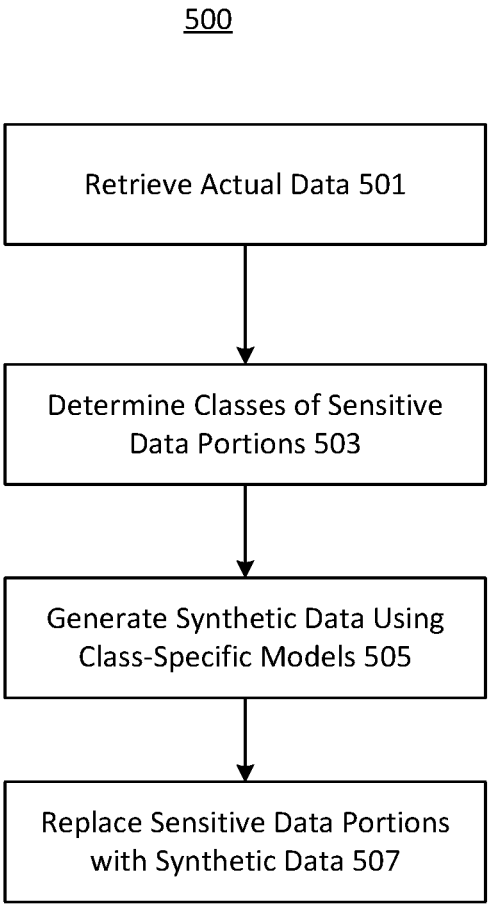

500
Retrieve Actual Data 501
Determine Classes of Sensitive Data Portions 503
Generate Synthetic Data Using Class-Specific Models 505
Replace Sensitive Data Portions with Synthetic Data 507

Fig. 5A

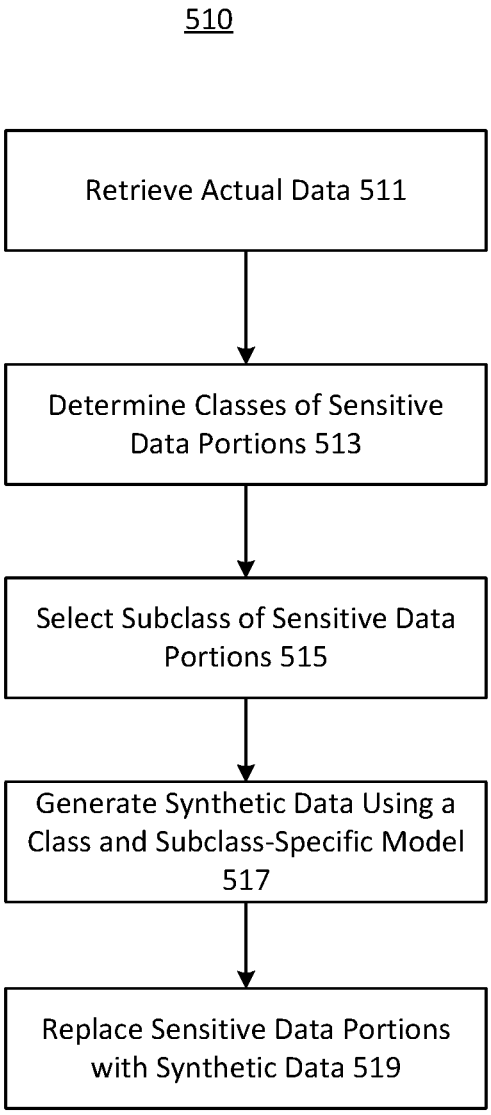

510
Retrieve Actual Data 511
Determine Classes of Sensitive Data Portions 513
Select Subclass of Sensitive Data Portions 515
Generate Synthetic Data Using a Class and Subclass-Specific Model 517
Replace Sensitive Data Portions with Synthetic Data 519

Fig. 5B

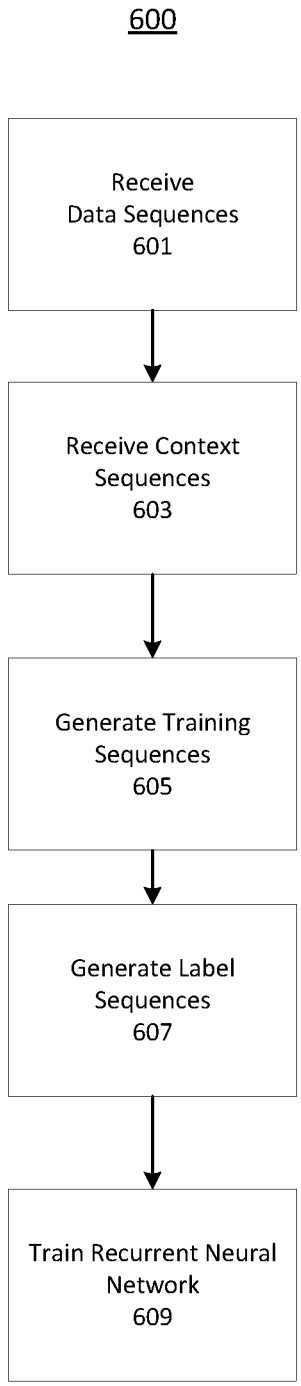

600
Receive
Data Sequences
601
Receive Context
Sequences
603
Generate Training
Sequences
605
Generate Label
Sequences
607
Train Recurrent Neural
Network
609

Fig. 6

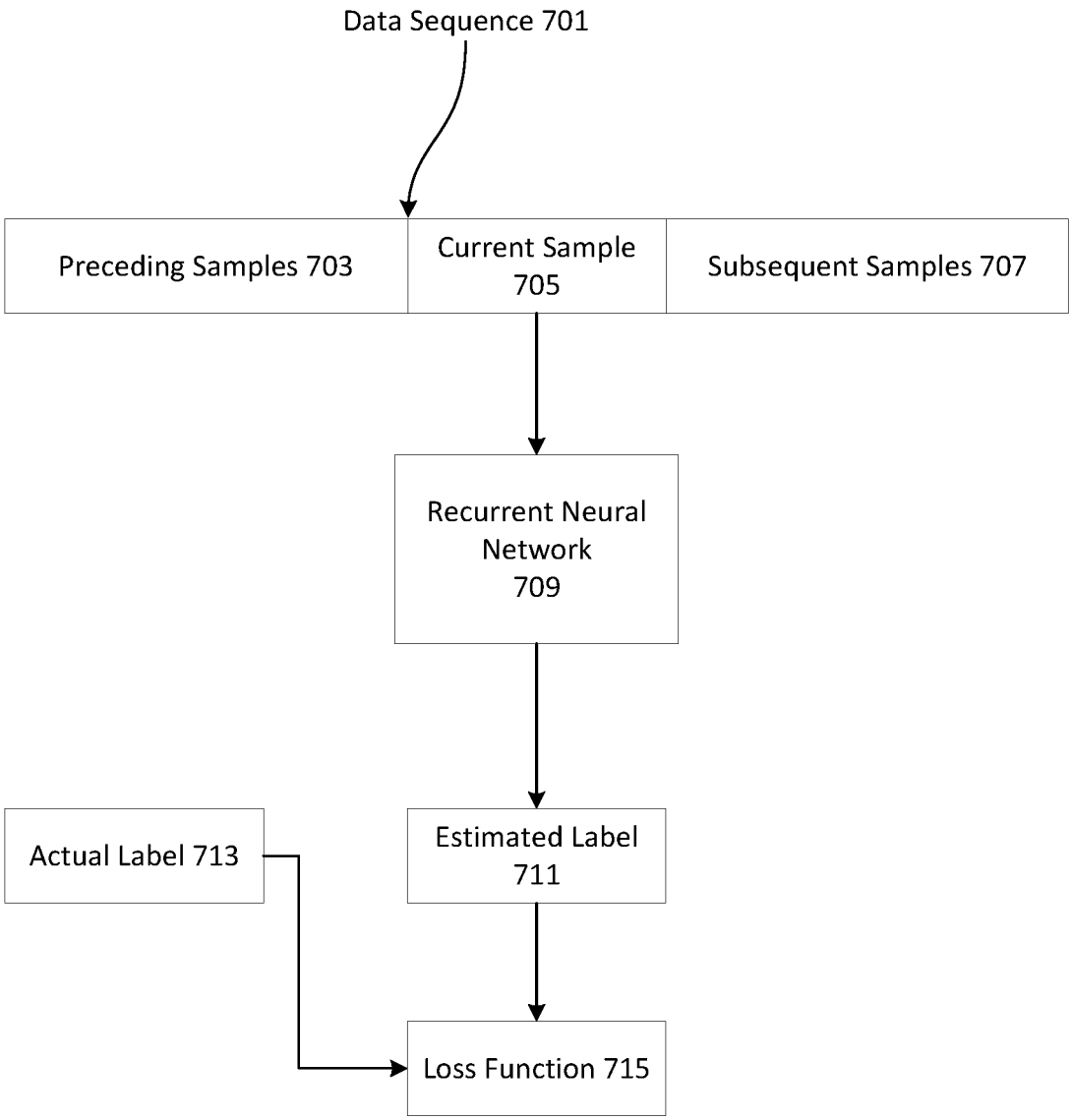

Data Sequence 701
Preceding Samples 703
Current Sample 705
Subsequent Samples 707
Recurrent Neural Network 709
Actual Label 713
Estimated Label 711
Loss Function 715

Fig. 7

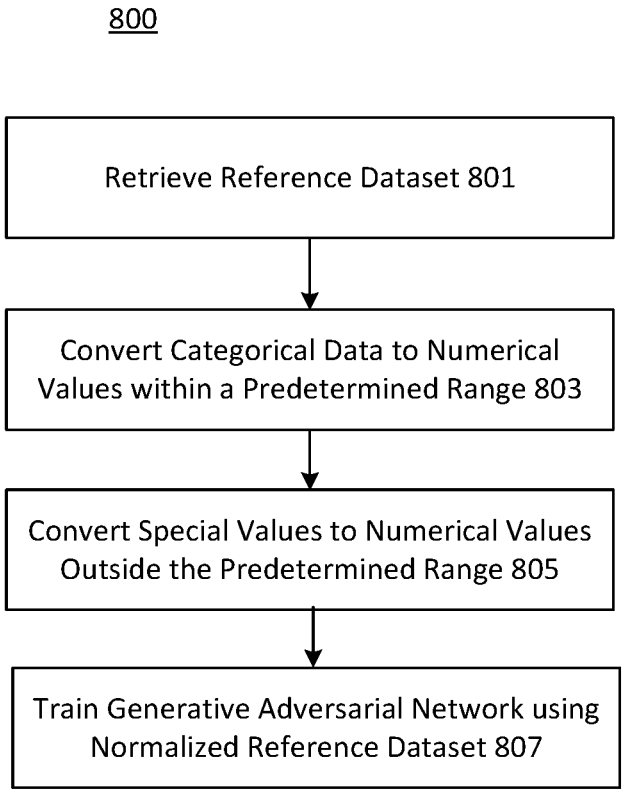

800
Retrieve Reference Dataset 801
Convert Categorical Data to Numerical Values within a Predetermined Range 803
Convert Special Values to Numerical Values Outside the Predetermined Range 805
Train Generative Adversarial Network using Normalized Reference Dataset 807

Fig. 8

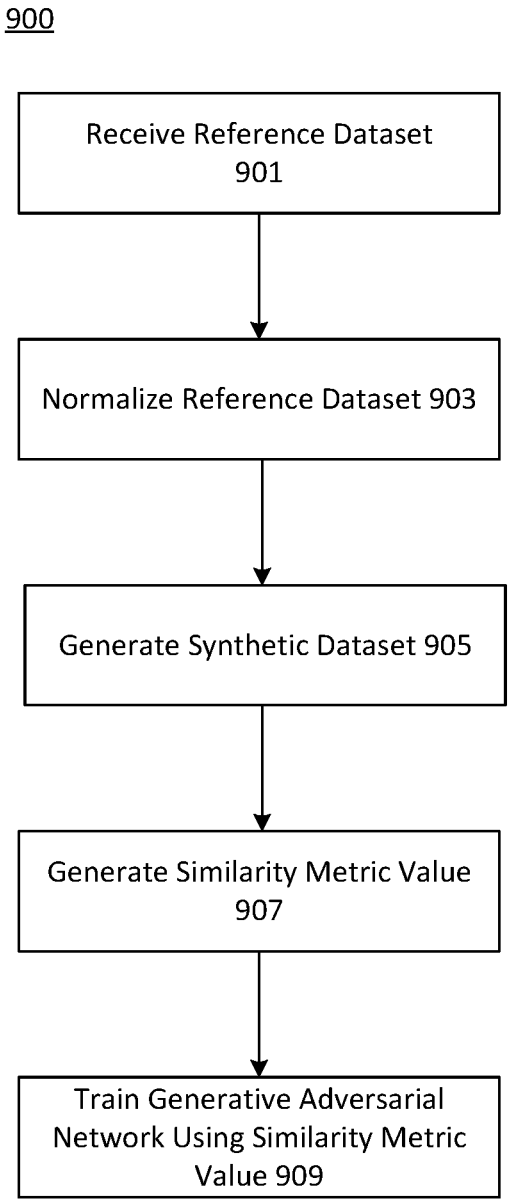

900
Receive Reference Dataset 901
Normalize Reference Dataset 903
Generate Synthetic Dataset 905
Generate Similarity Metric Value 907
Train Generative Adversarial Network Using Similarity Metric Value 909

Fig. 9

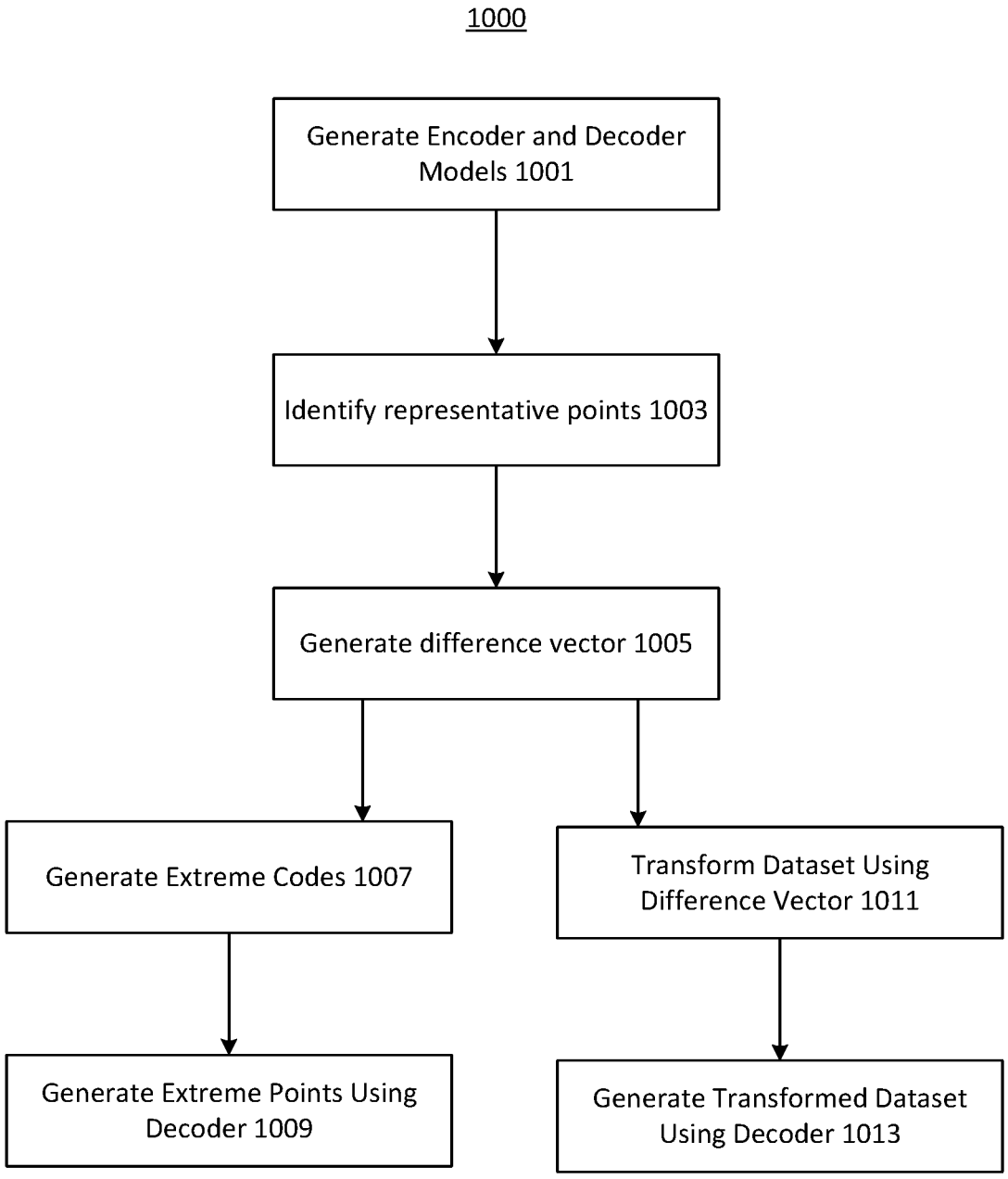

1000
Generate Encoder and Decoder Models 1001
Identify representative points 1003
Generate difference vector 1005
Generate Extreme Codes 1007
Transform Dataset Using Difference Vector 1011
Generate Extreme Points Using Decoder 1009
Generate Transformed Dataset Using Decoder 1013

Fig. 10

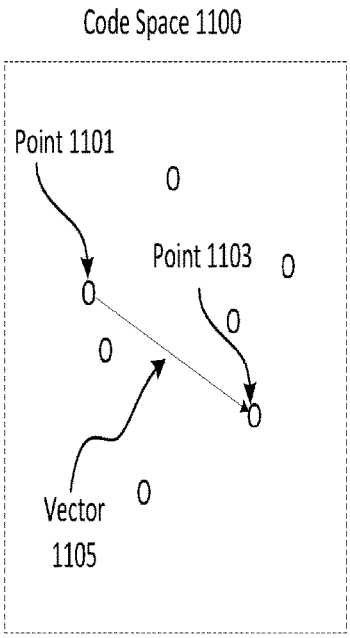


Fig. 11A

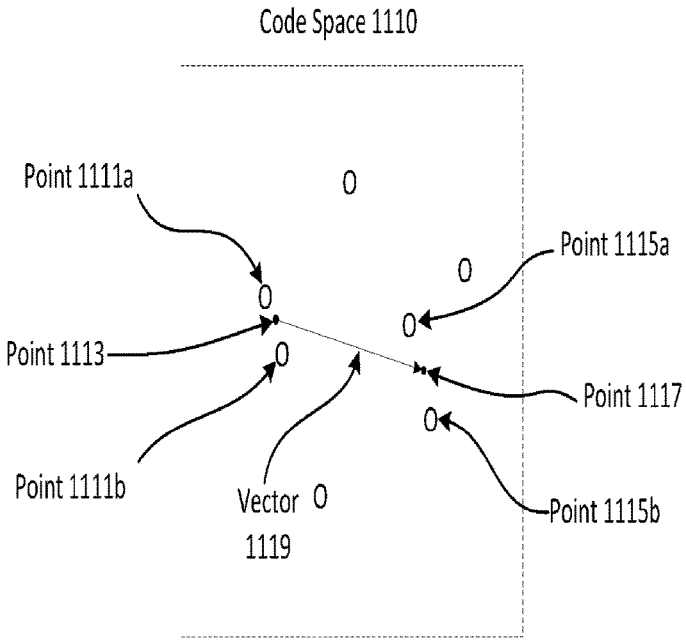


Fig. 11B

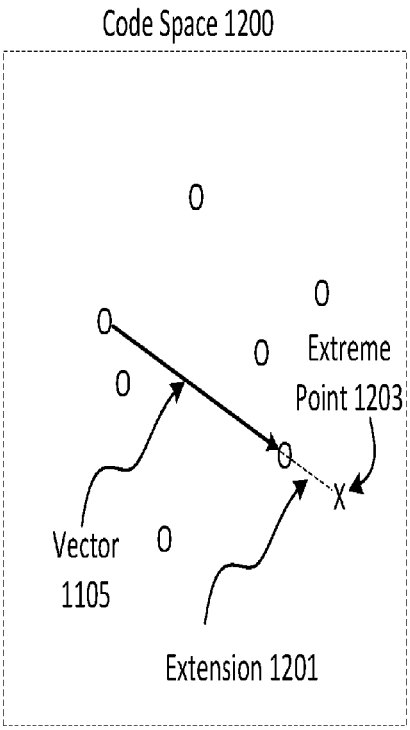


Fig. 12A

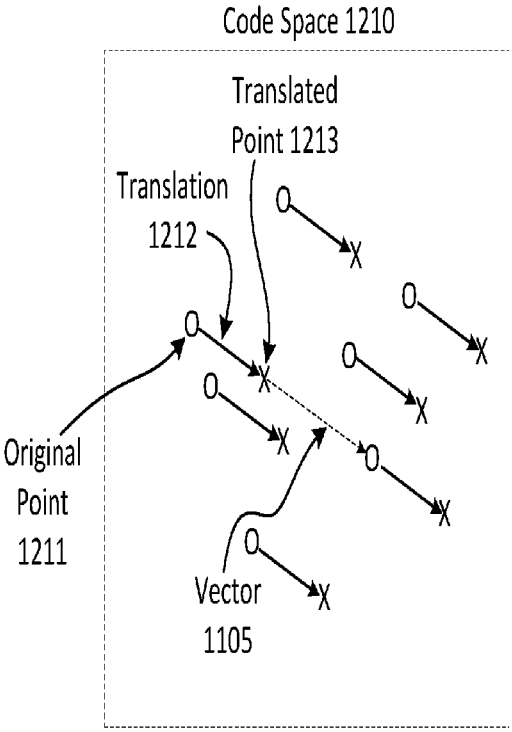


Fig. 12B

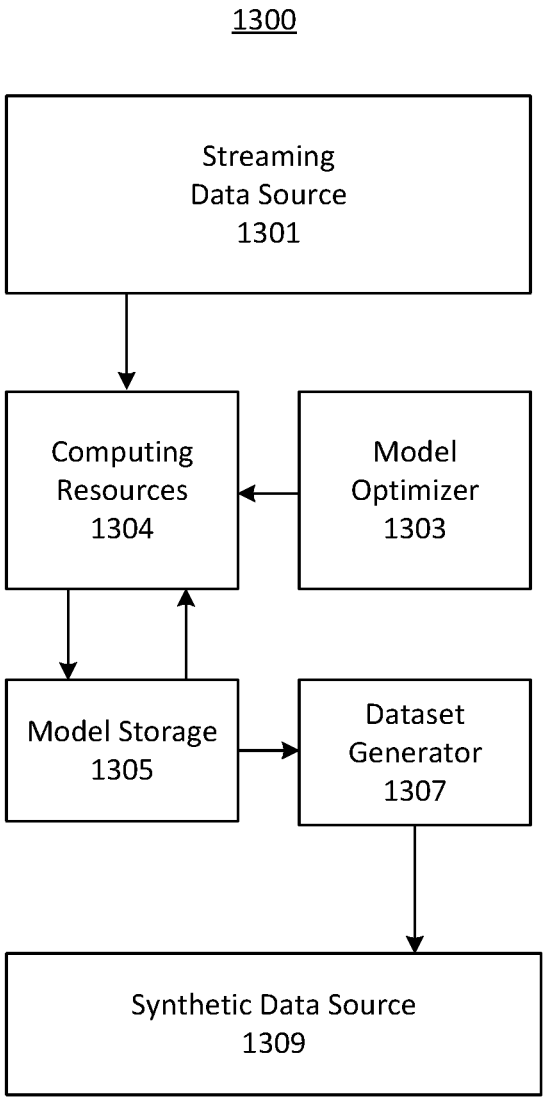

1300
Streaming
Data Source
1301
Computing
Resources
1304
Model
Optimizer
1303
Model Storage
1305
Dataset
Generator
1307
Synthetic Data Source
1309

Fig. 13

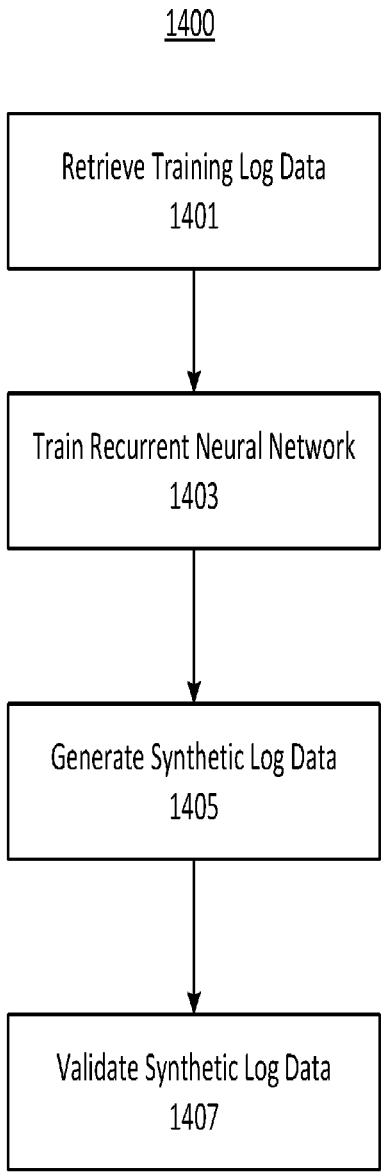

1400
Retrieve Training Log Data
1401
Train Recurrent Neural Network
1403
Generate Synthetic Log Data
1405
Validate Synthetic Log Data
1407

Fig. 14

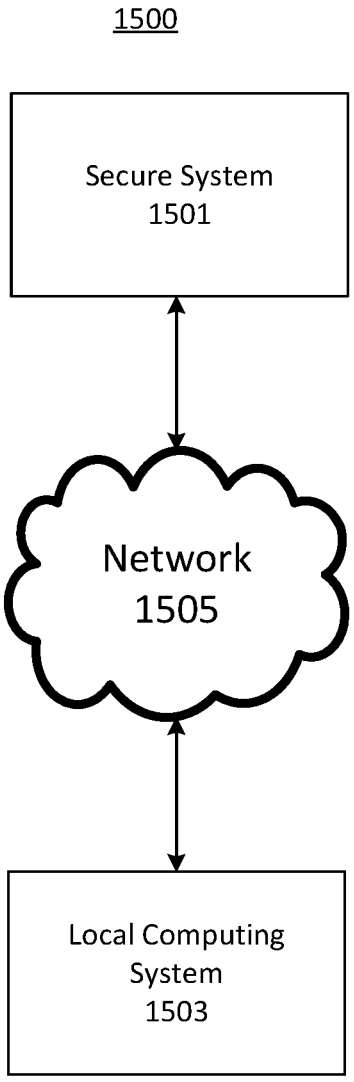

1500
Secure System
1501
Network
1505
Local Computing
System
1503

Fig. 15

SYSTEMS AND METHODS FOR SYNTHETIC DATA GENERATION

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/694,968, filed Jul. 6, 2018, and incorporated herein by reference in its entirety.

[0002] This application also relates to U.S. patent application Ser. No. _____, (Attorney Docket No. 2951/279202) filed on Oct. 4, 2018, and titled System, Method, and Computer-Accessible Medium for Evaluating Multi-Dimensional Synthetic Data Using Integrated Variants Analysis, the disclosure of which is also incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0003] The disclosed embodiments concern a platform for management of artificial intelligence systems. In particular, the disclosed embodiments concern using the disclosed platform to create models of data. These data models can be used to generate synthetic data for testing or training artificial intelligence systems. The disclosed embodiments also concern improvements to generative adversarial network models and adversarially learned inference models.

BACKGROUND

[0004] Training artificial intelligence systems can require substantial amounts of training data. Furthermore, when used with data dissimilar from the training data, artificial intelligence systems may perform poorly. These characteristics can create problems for developers of artificial intelligence applications designed to operate on sensitive data, such as customer financial records or patient healthcare data. Regulations governing the storage, transmission, and distribution of such data can inhibit application development, by forcing the development environment to comply with these burdensome regulations.

[0005] Furthermore, synthetic data can be generally useful for testing applications and systems. However, existing methods of creating synthetic data can be extremely slow and error-prone. For example, attempts to automatically desensitize data using regular expressions or similar methods requires substantial expertise and can fail when sensitive data is present in unanticipated formats or locations. Manual attempts to desensitize data can fall victim to human error. Neither approach will create synthetic data having statistical characteristics similar to those of the original data, limiting the utility of such data for training and testing purposes.

[0006] Accordingly, a need exists for systems and methods of creating synthetic data similar to existing datasets.

SUMMARY

[0007] The disclosed embodiments can improve generation of machine learning models. Such models may perform better on data similar to the data used to train them. But sensitive data cannot be widely distributed for use in training models, forcing application developers to choose between accuracy and training data security. Furthermore, the security of sensitive data can be improved by tokenizing sensitive data. For example, such tokenization can result in tokenized data, sensitive data values, and a mapping between the tokens and the values. An attacker must obtain both the tokenized data, sensitive data values, and mapping to reconstruct the sensitive data. But the process of manually tokenizing data is slow and error prone. The disclosed embodiments describe specific ways to generate synthetic data similar to sensitive data and to generate data models for tokenizing sensitive data. In this manner, the disclosed embodiments improve upon existing methods by enabling automatic generation of synthetic data and automatic tokenization of sensitive portions of datasets.

[0008] The disclosed embodiments may include a method for generating data models. The method can include receiving a data model generation request. The request can be received by a model optimizer from an interface. The method can include provisioning, by the model optimizer, computing resources with a data model. Then a dataset generator can generate a synthetic dataset for training the data model using a generative network of a generative adversarial network. The generative network can be trained to generate output data differing at least a predetermined amount from a reference dataset according to a similarity metric. The computing resources can train the data model using the synthetic dataset. The model optimizer can evaluate performance criteria of the data model. The model optimizer can store the data model and metadata of the data model in a model storage based on the evaluation of the performance criteria of the data model. Production data can then be processed using the trained data model.

[0009] The similarity metric can depend on a maximum distance or an average distance according to a distance measure between rows selected from the output data and row selected from the reference dataset.

[0010] The generative network can be trained to generate the output data with less than a predetermined proportion of duplicate elements. The generative network can be trained to generate the output data with an output data schema matching a schema of the reference dataset. The method can further comprise training the generative adversarial network using a loss function that penalizes generation of data differing from the reference dataset by less than the predetermined amount.

[0011] The model optimizer can be configured to generate at least one of a statistical correlation score between the synthetic dataset and the reference dataset, a data similarity score between the synthetic dataset and the reference dataset, and a data quality score for the synthetic dataset. Training the data model using the synthetic dataset can include determining that the synthetic dataset satisfies a criterion concerning the at least one of the statistical correlation score between the synthetic dataset and the reference dataset, the data similarity score between the synthetic dataset and the reference dataset, and the data quality score for the synthetic dataset.

[0012] The method can further include receiving the reference dataset, normalizing the reference dataset, and generating a synthetic training dataset using the generative network. A similarity metric value can be determined according to the similarity metric using the normalized reference dataset and the synthetic training dataset. A loss function can be updated that penalizes generation of data differing from the reference dataset by less than the predetermined amount using the similarity metric value. The generative adversarial network can be trained using the normalized reference dataset, the synthetic training dataset, and the updated loss function.

[0013] The generative network can include a decoder network configured to generate decoder output data in a sample space having a first dimensionality from decoder input data in a code space having a second dimensionality less than the first dimensionality. Generating the synthetic dataset for training the data model can include identifying first points and second points in the sample space and generating first corresponding points and second corresponding points in the code space using an encoder network corresponding to the decoder network, the first points, and the second points. Generating the synthetic dataset for training the data model can further include determining a first representative point based on the first corresponding points and a second representative point based on the second corresponding points and determining a vector connecting the first representative point and the second representative point. Datapoints in the code space can be translated using the vector and a scaling factor, and the translated datapoints can be converted into the sample space using the decoder network. The first representative point can be a centroid or a medoid of the first corresponding points.

[0014] Generating the synthetic dataset for training the data model can include identifying a first point and a second point in the sample space. Generating the synthetic dataset for training the data model can further include generating a first representative point and a second representative point in the code space using the first point, the second point, and an encoder network corresponding to the decoder network. A vector can be determined connecting the first representative point and the second representative point. An extreme point in the code space can be generated by sampling the code space along an extension of the vector beyond the second representative point. The extreme point in the code space can be converted into the sample space using the decoder network.

[0015] The disclosed embodiments may include a cloud computing system for generating data models. The cloud computing system can include at least one processor and at least one non-transitory memory storing instructions that, when executed by the at least one processor, cause the cloud computing system to perform the following operations. A model optimizer can receive, from an interface a data model generation request. The model optimizer can provision computing resources with a data model. A generative network of a generative adversarial network can generate a synthetic dataset for training the data model. The computing resources can train the data model using the synthetic dataset. The model optimizer can evaluate the performance criteria of the data model. The model optimizer can store, in a model storage, the data model and metadata of the data model based on the evaluation of the performance criteria of the data model. Production data can then be processed using the data model.

[0016] In some aspects, the operations can further include retrieving a reference dataset from a database, the reference dataset including categorical data. A normalized training dataset can be generated by normalizing the categorical data. The generative network can be trained using the normalized training dataset. Normalizing the categorical data can include converting the categorical data to numerical values within a predetermined range. The reference dataset can include at least one of missing values or not-a-number values. Generating the normalized training dataset by normalizing the categorical data can include converting the at least one of the missing values or the not-a-number values to corresponding predetermined numerical values outside the predetermined range.

[0017] The generative network can be configured to generate output data differing at least a predetermined amount from a reference dataset according to a similarity metric. In some aspects, the similarity metric can depend on a covariance of numeric elements of the output data and a covariance of numeric elements of the reference dataset. In various aspects, the similarity metric can depend on a univariate value distribution of an element of the output data and a univariate value distribution of an element of the reference dataset. The similarity metric can depend on a joint probability distribution of elements in the output data and a joint probability distribution of elements in the reference dataset. The similarity metric can depend on a number of rows of the output data that match rows of the reference dataset.

[0018] The disclosed embodiments may include a cloud computing system for generating data models. The cloud computing system can include at least one processor and at least one non-transitory memory storing instructions that, when executed by the at least one processor cause the cloud computing system to perform the following operations. A model optimizer can provision computing resources with a data model. A dataset generator can generate a synthetic dataset for training the data model. The computing resources can train the data model using the synthetic dataset. The model optimizer can store, in a model storage, the data model and metadata of the data model. Production data can be received from a data source by a production instance using a common file system. The production data can be processed using the data model by the production instance. The computing resources, the dataset generator, and the model optimizer can be hosted by separate virtual computing instances of the cloud computing system. A can distributor routes requests between the computing resources, the dataset generator, and the model optimizer. The data model can be provisioned in response to a model generation request received by the model optimizer from an interface. The model optimizer can evaluate performance criteria of the data model. The performance criteria can include at least one of a statistical correlation score, data similarity score, or data quality score, prediction accuracy check, a prediction accuracy cross check, a regression check, a regression cross check, and a principal component analysis check. A model curator can determine that the data model satisfies governance criteria.

[0019] Generating the synthetic dataset for training the data model can include retrieving a synthetic dataset model from the model storage, retrieving a training dataset from a database, and generating the synthetic dataset using the synthetic dataset model and the training dataset. Generating the synthetic dataset using the synthetic dataset model and the training dataset can include identifying a sensitive portion of the training dataset using a recurrent neural network. In some aspects, the cloud computing system can perform further operations of receiving a data sequence and receiving a context sequence. The operations can additionally include generating a training sequence by inserting the data sequence into the context sequence and generating a label sequence indicating a position of the inserted data sequence in the training sequence. The recurrent neural network can be trained using the training sequence and the label

sequence. When the training sequence includes inserted data sequences, the label sequence can indicate at least one of differing classes among the inserted data sequences and differing subclasses among the inserted data sequences.

[0020] In some aspects, training the recurrent neural network using the training sequence and the label sequence can include estimating a label by applying a subset of the training sequence to the recurrent neural network and comparing the estimated label to an actual label in the label sequence, the actual label corresponding to the subset of the training sequence. The recurrent neural network can be updated according to a loss function based on a result of the comparison. The actual label can correspond to an element of the subset occupying the same position in the training sequence as the actual label occupies in the label sequence.

[0021] The disclosed embodiments may include a method for generating data models. According to the method, a model optimizer can receive from an interface, a data model generation request. The model optimizer can provision computing resources with the data model. A dataset generator can generate a synthetic dataset for training the data model. The computing resources can train the data model using the synthetic dataset. The model optimizer can determine metadata of the data model. The model optimizer can store, in a model storage, the data model and metadata of the data model. The production data can be processed using the data model. A model curator can determine that the data model satisfies governance criteria, before processing the production data using the data model. In some aspects, the interface, the computing resources, the dataset generator, and the model optimizer can be hosted by separate virtual computing instances of a cloud computing system. A distributor can route user requests to the computing resources, the dataset generator, and the model optimizer. The production data can be received from a data source by a production instance using a common file system. The production data can be processed using the data model by the production instance.

[0022] Generating the synthetic dataset for training the data model can include retrieving a synthetic dataset model from the model storage and retrieving a training dataset from a database. The synthetic dataset can be generated using the synthetic dataset model and the training dataset.

[0023] In some aspects, the synthetic dataset model can include a class-specific model corresponding to a data class. Generating the synthetic dataset using the synthetic dataset model and the training dataset can include determining a sensitive portion of the training dataset belongs to the data class and generating a synthetic portion using the class-specific model. The sensitive portion of the training dataset can then be replaced with the synthetic portion.

[0024] In various aspects, the synthetic dataset model can include a class and subclass-specific model corresponding to a data class and a subclass of the data class. Generating the synthetic dataset using the synthetic dataset model and the training dataset can include determining a sensitive portion of the training dataset belongs to the data class, selecting the subclass and generating a synthetic portion using the class and subclass-specific model. The sensitive portion of the training dataset can then be replaced with the synthetic portion. The subclass can be selected according to a univariate distribution, or using a recurrent neural network.

[0025] A non-transitory memory can store instructions that, when executed by at least one processor, cause a system to perform operations of obtaining a synthetic dataset model, retrieving a training dataset from a database; and generating a synthetic dataset using the synthetic dataset model and the training data. This generation can include determining a sensitive portion of the training dataset belongs to a data class using a recurrent neural network, selecting a data subclass according to a univariate distribution, and generating a synthetic portion using a class and subclass-specific model. The sensitive portion of the training dataset can then be replaced with the synthetic portion.

[0026] The disclosed embodiments may include a cloud computing system for generating a synthetic data stream. The cloud computing system can include at least one processor and at least one non-transitory memory storing instructions that, when executed by the at least one processor cause the cloud computing system to perform the following operations. A model optimizer can receive a synthetic data stream request indicating a reference data stream from an interface. A dataset generator can generate a synthetic data stream that tracks the reference data stream by repeatedly swapping data models of the reference data stream. One such repeat can include the following steps. The dataset generator can retrieve a current data model of the reference data stream from a model storage. In some aspects, the current data model comprises at least one of a kernel density estimator, a recurrent neural network, and a generative adversarial network. The dataset generator can generate the synthetic data stream using the current data model of the reference data stream. A new data model of the reference data stream can be generated, and the model optimizer can store the new data model in the model storage. Generating the new data model of the reference data stream can include provisioning, by the model optimizer, computing resources with the current data model training the new data model on the computing resources using current reference data stream data.

[0027] The repeat can further include the steps of receiving reference data stream data. In some aspects, the reference data stream data can be included into current reference data stream data upon receipt. In various aspects, the received reference data stream data can be stored, reference data stream data stored during a previous repeat can be retrieved, and the retrieved reference data stream data can be included into the current reference data stream data. The repeat can occur at a predetermined time or upon expiration of a time interval. The repeat can occur when a data schema of the reference data stream changes. In some aspect, the repeat can include evaluating, by the model optimizer, performance criteria of the new data model and determining, by the model optimizer, metadata of the new data model. The new data model and the metadata can then be stored based on the evaluation of the performance criteria of the new data model. The performance criteria can include at least one of a statistical correlation score, data similarity score, or data quality score, prediction accuracy check, a prediction accuracy cross check, a regression check, a regression cross check, and a principal component analysis check.

[0028] In some aspects, the data models can comprise recurrent neural networks and the reference data stream comprises JSON log data. Generating the synthetic data stream using the current data model of the reference data stream can include validating the synthetic data stream using a JSON validator and a schema for the reference data stream. In various aspects, the schema can describe key-value pairs

present in the reference data stream, and validating the synthetic data stream can include validating that keys present in the synthetic data stream are present in the schema. In some aspects, the schema can describe key-value pairs present in the reference data stream, and validating the synthetic data stream can include validating that key-value formats present in the synthetic data stream match corresponding key-value formats in the reference data stream.

[0029] Generating the synthetic data stream using the current data model of the reference data stream can include identifying a sensitive portion of the reference data stream using a recurrent neural network and generating a synthetic portion using the current data model. The sensitive portion of the reference data stream can then be replaced with the synthetic portion. In some aspects, the current data model can include a class-specific model corresponding to a data class. Identifying the sensitive portion of the reference data stream can include determining the sensitive portion of the reference data stream belongs to the data class. Generating the synthetic portion can include selecting the class-specific model based on the data class and generating the synthetic portion using the class-specific model. In various aspects, the current data model can include a class and subclass-specific model corresponding to a data class and a subclass of the data class. Identifying the sensitive portion of the reference data stream can include determining the sensitive portion of the reference data stream belongs to the data class. Generating the synthetic portion can include selecting the subclass and selecting the class and subclass-specific model based on the data class and the selected subclass. The synthetic portion can then be generated using the class and subclass-specific model.

[0030] The disclosed embodiments may include a system for generating data models. The system for generating data models can include at least one secure system processor and at least one secure system non-transitory memory storing first instructions that, when executed by the at least one secure system processor, cause a secure system to perform the following secure system operations. A model optimizer can receive, from an interface, a data model generation request. The model optimizer can provision the computing resources with a data model. The computing resources can train the data model using a sensitive dataset. The model optimizer can then store the data model in a model storage. The system for generating data models can also include at least one insecure system processor and at least one insecure system non-transitory memory storing second instructions that, when executed by the at least one insecure system processor cause an insecure system to receive a data generation request, retrieve the data model from the secure system based on the data generation request and the metadata of the data model, and generating synthetic data using the data model in response to the data generation request. The data model can include at least one of a kernel density estimator, a recurrent neural network, and a generative adversarial network. When data model includes a generative adversarial network, the generative adversarial network can include a generative network. This generative network can be trained to generate output data differing at least a predetermined amount from a reference dataset according to a similarity metric. The secure system can be a cloud computing system. The insecure system can be a personal computer or mobile device.

[0031] It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the disclosed embodiments, as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0032] The drawings are not necessarily to scale or exhaustive. Instead, emphasis is generally placed upon illustrating the principles of the embodiments described herein. The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments consistent with the disclosure and, together with the description, serve to explain the principles of the disclosure. In the drawings:

[0033] FIG. **1** depicts an exemplary cloud-computing environment for generating data models, consistent with disclosed embodiments.

[0034] FIG. **2** depicts an exemplary process for generating data models, consistent with disclosed embodiments.

[0035] FIG. **3** depicts an exemplary process for generating synthetic data using existing data models, consistent with disclosed embodiments.

[0036] FIG. **4** depicts an exemplary implementation of the cloud-computing environment of FIG. **1**, consistent with disclosed embodiments.

[0037] FIG. **5**A depicts an exemplary process for generating synthetic data using class-specific models, consistent with disclosed embodiments.

[0038] FIG. **5**B depicts an exemplary process for generating synthetic data using class and subclass-specific models, consistent with disclosed embodiments.

[0039] FIG. **6** depicts an exemplary process for training a classifier for generation of synthetic data, consistent with disclosed embodiments.

[0040] FIG. **7** depicts an exemplary process for training a classifier for generation of synthetic data, consistent with disclosed embodiments.

[0041] FIG. **8** depicts an exemplary process for training a generative adversarial using a normalized reference dataset, consistent with disclosed embodiments.

[0042] FIG. **9** depicts an exemplary process for training a generative adversarial network using a loss function configured to ensure a predetermined degree of similarity, consistent with disclosed embodiments.

[0043] FIG. **10** depicts an exemplary process for supplementing or transform datasets using code-space operations, consistent with disclosed embodiments.

[0044] FIGS. **11**A and **11**B depict an exemplary illustration of points in code-space, consistent with disclosed embodiments.

[0045] FIG. **12**A depicts an exemplary illustration of supplementing datasets using code-space operations, consistent with disclosed embodiments.

[0046] FIG. **12**B depicts an exemplary illustration of transforming datasets using code-space operations, consistent with disclosed embodiments.

[0047] FIG. **13** depicts an exemplary cloud computing system for generating a synthetic data stream that tracks a reference data stream, consistent with disclosed embodiments.

[0048] FIG. **14** depicts a process for generating synthetic JSON log data using the cloud computing system of FIG. **13**, consistent with disclosed embodiments.

[0049] FIG. **15** depicts a system for secure generation and insecure use of models of sensitive data, consistent with disclosed embodiments.

DETAILED DESCRIPTION

[0050] Reference will now be made in detail to exemplary embodiments, discussed with regards to the accompanying drawings. In some instances, the same reference numbers will be used throughout the drawings and the following description to refer to the same or like parts. Unless otherwise defined, technical and/or scientific terms have the meaning commonly understood by one of ordinary skill in the art. The disclosed embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosed embodiments. It is to be understood that other embodiments may be utilized and that changes may be made without departing from the scope of the disclosed embodiments. Thus, the materials, methods, and examples are illustrative only and are not intended to be necessarily limiting.

[0051] The disclosed embodiments can be used to create models of datasets, which may include sensitive datasets (e.g., customer financial information, patient healthcare information, and the like). Using these models, the disclosed embodiments can produce fully synthetic datasets with similar structure and statistics as the original sensitive or non-sensitive datasets. The disclosed embodiments also provide tools for desensitizing datasets and tokenizing sensitive values. In some embodiments, the disclosed systems can include a secure environment for training a model of sensitive data, and a non-secure environment for generating synthetic data with similar structure and statistics as the original sensitive data. In various embodiments, the disclosed systems can be used to tokenize the sensitive portions of a dataset (e.g., mailing addresses, social security numbers, email addresses, account numbers, demographic information, and the like). In some embodiments, the disclosed systems can be used to replace parts of sensitive portions of the dataset (e.g., preserve the first or last 3 digits of an account number, social security number, or the like; change a name to a first and last initial). In some aspects, the dataset can include one or more JSON (JavaScript Object Notation) or delimited files (e.g., comma-separated value, or CSV, files). In various embodiments, the disclosed systems can automatically detect sensitive portions of structured and unstructured datasets and automatically replace them with similar but synthetic values.

[0052] FIG. **1** depicts a cloud-computing environment **100** for generating data models. Environment **100** can be configured to support generation and storage of synthetic data, generation and storage of data models, optimized choice of parameters for machine learning, and imposition of rules on synthetic data and data models. Environment **100** can be configured to expose an interface for communication with other systems. Environment **100** can include computing resources **101**, dataset generator **103**, database **105**, model optimizer **107**, model storage **109**, model curator **111**, and interface **113**. These components of environment **100** can be configured to communicate with each other, or with external components of environment **100**, using network **115**. The particular arrangement of components depicted in FIG. **1** is not intended to be limiting. System **100** can include additional components, or fewer components. Multiple components of system **100** can be implemented using the same physical computing device or different physical computing devices.

[0053] Computing resources **101** can include one or more computing devices configurable to train data models. The computing devices can be special-purpose computing devices, such as graphical processing units (GPUs) or application-specific integrated circuits. The cloud computing instances can be general-purpose computing devices. The computing devices can be configured to host an environment for training data models. For example, the computing devices can host virtual machines, pods, or containers. The computing devices can be configured to run applications for generating data models. For example, the computing devices can be configured to run SAGEMAKER, GENESYS, or similar machine learning training applications. Computing resources **101** can be configured to receive models for training from model optimizer **107**, model storage **109**, or another component of system **100**. Computing resources **101** can be configured provide training results, including trained models and model information, such as the type and/or purpose of the model and any measures of classification error.

[0054] Dataset generator **103** can include one or more computing devices configured to generate data. Dataset generator **103** can be configured to provide data to computing resources **101**, database **105**, to another component of system **100** (e.g., interface **113**), or another system (e.g., an APACHE KAFKA cluster or other publication service). Dataset generator **103** can be configured to receive data from database **105** or another component of system **100**. Dataset generator **103** can be configured to receive data models from model storage **109** or another component of system **100**. Dataset generator **103** can be configured to generate synthetic data. For example, dataset generator **103** can be configured to generate synthetic data by identifying and replacing sensitive information in data received from database **103** or interface **113**. As an additional example, dataset generator **103** can be configured to generate synthetic data using a data model without reliance on input data. For example, the data model can be configured to generate data matching statistical and content characteristics of a training dataset. In some aspects, the data model can be configured to map from a random or pseudorandom vector to elements in the training data space.

[0055] Database **105** can include one or more databases configured to store data for use by system **100**. The databases can include cloud-based databases (e.g., AMAZON WEB SERVICES S3 buckets) or on-premises databases.

[0056] Model optimizer **107** can include one or more computing systems configured to manage training of data models for system **100**. Model optimizer **107** can be configured to generate models for export to computing resources **101**. Model optimizer **107** can be configured to generate models based on instructions received from a user or another system. These instructions can be received through interface **113**. For example, model optimizer **107** can be configured to receive a graphical depiction of a machine learning model and parse that graphical depiction into instructions for creating and training a corresponding neural network on computing resources **101**. Model optimizer **107** can be configured to select model training parameters. This selection can be based on model performance feedback received from computing resources **101**. Model

optimizer **107** can be configured to provide trained models and descriptive information concerning the trained models to model storage **109**.

[0057] Model storage **109** can include one or more databases configured to store data models and descriptive information for the data models. Model storage **109** can be configured to provide information regarding available data models to a user or another system. This information can be provided using interface **113**. The databases can include cloud-based databases (e.g., AMAZON WEB SERVICES S3 buckets) or on-premises databases. The information can include model information, such as the type and/or purpose of the model and any measures of classification error.

[0058] Model curator **111** can be configured to impose governance criteria on the use of data models. For example, model curator **111** can be configured to delete or control access to models that fail to meet accuracy criteria. As a further example, model curator **111** can be configured to limit the use of a model to a particular purpose, or by a particular entity or individual. In some aspects, model curator **11** can be configured to ensure that data model satisfies governance criteria before system **100** can process data using the data model.

[0059] Interface **113** can be configured to manage interactions between system **100** and other systems using network **115**. In some aspects, interface **113** can be configured to publish data received from other components of system **100** (e.g., dataset generator **103**, computing resources **101**, database **105**, or the like). This data can be published in a publication and subscription framework (e.g., using APACHE KAFKA), through a network socket, in response to queries from other systems, or using other known methods. The data can be synthetic data, as described herein. As an additional example, interface **113** can be configured to provide information received from model storage **109** regarding available datasets. In various aspects, interface **113** can be configured to provide data or instructions received from other systems to components of system **100**. For example, interface **113** can be configured to receive instructions for generating data models (e.g., type of data model, data model parameters, training data indicators, training parameters, or the like) from another system and provide this information to model optimizer **107**. As an additional example, interface **113** can be configured to receive data including sensitive portions from another system (e.g. in a file, a message in a publication and subscription framework, a network socket, or the like) and provide that data to dataset generator **103** or database **105**.

[0060] Network **115** can include any combination of electronics communications networks enabling communication between components of system **100**. For example, network **115** may include the Internet and/or any type of wide area network, an intranet, a metropolitan area network, a local area network (LAN), a wireless network, a cellular communications network, a Bluetooth network, a radio network, a device bus, or any other type of electronics communications network known to one of skill in the art.

[0061] FIG. **2** depicts a process **200** for generating data models. Process **200** can be used to generate a data model for a machine learning application, consistent with disclosed embodiments. The data model can be generated using synthetic data in some aspects. This synthetic data can be generated using a synthetic dataset model, which can in turn be generated using actual data. The synthetic data may be similar to the actual data in terms of values, value distributions (e.g., univariate and multivariate statistics of the synthetic data may be similar to that of the actual data), structure and ordering, or the like. In this manner, the data model for the machine learning application can be generated without directly using the actual data. As the actual data may include sensitive information, and generating the data model may require distribution and/or review of training data, the use of the synthetic data can protect the privacy and security of the entities and/or individuals whose activities are recorded by the actual data.

[0062] Process **200** can then proceed to step **201**. In step **201**, interface **113** can provide a data model generation request to model optimizer **107**. The data model generation request can include data and/or instructions describing the type of data model to be generated. For example, the data model generation request can specify a general type of data model (e.g., neural network, recurrent neural network, generative adversarial network, kernel density estimator, random data generator, or the like) and parameters specific to the particular type of model (e.g., the number of features and number of layers in a generative adversarial network or recurrent neural network). In some embodiments, a recurrent neural network can include long short term memory modules (LSTM units), or the like.

[0063] Process **200** can then proceed to step **203**. In step **203**, one or more components of system **100** can interoperate to generate a data model. For example, as described in greater detail with regard to FIG. **3**, a data model can be trained using computing resources **101** using data provided by dataset generator **103**. In some aspects, this data can be generated using dataset generator **103** from data stored in database **105**. In various aspects, the data used to train dataset generator **103** can be actual or synthetic data retrieved from database **105**. This training can be supervised by model optimizer **107**, which can be configured to select model parameters (e.g., number of layers for a neural network, kernel function for a kernel density estimator, or the like), update training parameters, and evaluate model characteristics (e.g., the similarity of the synthetic data generated by the model to the actual data). In some embodiments, model optimizer **107** can be configured to provision computing resources **101** with an initialized data model for training. The initialized data model can be, or can be based upon, a model retrieved from model storage **109**.

[0064] Process **200** can then proceed to step **205**. In step **205**, model optimizer **107** can evaluate the performance of the trained synthetic data model. When the performance of the trained synthetic data model satisfies performance criteria, model optimizer **107** can be configured to store the trained synthetic data model in model storage **109**. For example, model optimizer **107** can be configured to determine one or more values for similarity and/or predictive accuracy metrics, as described herein. In some embodiments, based on values for similarity metrics, model optimizer **107** can be configured to assign a category to the synthetic data model.

[0065] According to a first category, the synthetic data model generates data maintaining a moderate level of correlation or similarity with the original data, matches well with the original schema, and does not generate too many row or value duplicates. According to a second category, the synthetic data model may generate data maintaining a high level of correlation or similarity of the original level, and

therefore could potentially cause the original data to be discernable from the original data (e.g., a data leak). A synthetic data model generating data failing to match the schema with the original data or providing many duplicated rows and values may also be placed in this category. According to a third category, the synthetic data model may likely generate data maintaining a high level of correlation or similarity with the original data, likely allowing a data leak. A synthetic data model generating data badly failing to match the schema with the original data or providing far too many duplicated rows and values may also be placed in this category.

[0066] In some embodiments, system **100** can be configured to provide instructions for improving the quality of the synthetic data model. If a user requires synthetic data reflecting less correlation or similarity with the original data, the use can change the models' parameters to make them perform worse (e.g., by decreasing number of layers in GAN models, or reducing the number of training iterations). If the users want the synthetic data to have better quality, they can change the models' parameters to make them perform better (e.g., by increasing number of layers in GAN models, or increasing the number of training iterations).

[0067] Process **200** can then proceed to step **207**, in step **207**, model curator **111** can evaluate the trained synthetic data model for compliance with governance criteria.

[0068] FIG. **3** depicts a process **300** for generating a data model using an existing synthetic data model, consistent with disclosed embodiments. Process **300** can include the steps of retrieving a synthetic dataset model from model storage **109**, retrieving data from database **105**, providing synthetic data to computing resources **101**, providing an initialized data model to computing resources **101**, and providing a trained data model to model optimizer **107**. In this manner, process **300** can allow system **100** to generate a model using synthetic data.

[0069] Process **300** can then proceed to step **301**. In step **301**, dataset generator **103** can retrieve a training dataset from database **105**. The training dataset can include actual training data, in some aspects. The training dataset can include synthetic training data, in some aspects. In some embodiments, dataset generator **103** can be configured to generate synthetic data from sample values. For example, dataset generator **103** can be configured to use the generative network of a generative adversarial network to generate data samples from random-valued vectors. In such embodiments, process **300** may forgo step **301**.

[0070] Process **300** can then proceed to step **303**. In step **303**, dataset generator **103** can be configured to receive a synthetic data model from model storage **109**. In some embodiments, model storage **109** can be configured to provide the synthetic data model to dataset generator **103** in response to a request from dataset generator **103**. In various embodiments, model storage **109** can be configured to provide the synthetic data model to dataset generator **103** in response to a request from model optimizer **107**, or another component of system **100**. As a non-limiting example, the synthetic data model can be a neural network, recurrent neural network (which may include LSTM units), generative adversarial network, kernel density estimator, random value generator, or the like.

[0071] Process **300** can then proceed to step **305**. In step **305**, in some embodiments, dataset generator **103** can generate synthetic data. Dataset generator **103** can be configured, in some embodiments, to identify sensitive data items (e.g., account numbers, social security numbers, names, addresses, API keys, network or IP addresses, or the like) in the data received from model storage **109**. In some embodiments, dataset generator **103** can be configured to identify sensitive data items using a recurrent neural network. Dataset generator **103** can be configured to use the data model retrieved from model storage **109** to generate a synthetic dataset by replacing the sensitive data items with synthetic data items.

[0072] Dataset generator **103** can be configured to provide the synthetic dataset to computing resources **101**. In some embodiments, dataset generator **103** can be configured to provide the synthetic dataset to computing resources **101** in response to a request from computing resources **101**, model optimizer **107**, or another component of system **100**. In various embodiments, dataset generator **103** can be configured to provide the synthetic dataset to database **105** for storage. In such embodiments, computing resources **101** can be configured to subsequently retrieve the synthetic dataset from database **105** directly, or indirectly through model optimizer **107** or dataset generator **103**.

[0073] Process **300** can then proceed to step **307**. In step **307**, computing resources **101** can be configured to receive a data model from model optimizer **107**, consistent with disclosed embodiments. In some embodiments, the data model can be at least partially initialized by model optimizer **107**. For example, at least some of the initial weights and offsets of a neural network model received by computing resources **101** in step **307** can be set by model optimizer **107**. In various embodiments, computing resources **101** can be configured to receive at least some training parameters from model optimizer **107** (e.g., batch size, number of training batches, number of epochs, chunk size, time window, input noise dimension, or the like).

[0074] Process **300** can then proceed to step **309**. In step **309**, computing resources **101** can generate a trained data model using the data model received from model optimizer **107** and the synthetic dataset received from dataset generator **103**. For example, computing resources **101** can be configured to train the data model received from model optimizer **107** until some training criterion is satisfied. The training criterion can be, for example, a performance criterion (e.g., a Mean Absolute Error, Root Mean Squared Error, percent good classification, and the like), a convergence criterion (e.g., a minimum required improvement of a performance criterion over iterations or over time, a minimum required change in model parameters over iterations or over time), elapsed time or number of iterations, or the like. In some embodiments, the performance criterion can be a threshold value for a similarity metric or prediction accuracy metric as described herein. Satisfaction of the training criterion can be determined by one or more of computing resources **101** and model optimizer **107**. In some embodiments, computing resources **101** can be configured to update model optimizer **107** regarding the training status of the data model. For example, computing resources **101** can be configured to provide the current parameters of the data model and/or current performance criteria of the data model. In some embodiments, model optimizer **107** can be configured to stop the training of the data model by computing resources **101**. In various embodiments, model optimizer **107** can be configured to retrieve the data model from computing resources **101**. In some embodiments, computing

resources **101** can be configured to stop training the data model and provide the trained data model to model optimizer **107**.

[0075] FIG. **4** depicts a specific implementation (system **400**) of system **100** of FIG. **1**. As shown in FIG. **4**, the functionality of system **100** can be divided between a distributor **401**, a dataset generation instance **403**, a development environment **405**, a model optimization instance **409**, and a production environment **411**. In this manner, system **100** can be implemented in a stable and scalable fashion using a distributed computing environment, such as a public cloud-computing environment, a private cloud computing environment, a hybrid cloud computing environment, a computing cluster or grid, or the like. As present computing requirements increase for a component of system **400** (e.g., as production environment **411** is called upon to instantiate additional production instances to address requests for additional synthetic data streams), additional physical or virtual machines can be recruited to that component. In some embodiments, dataset generator **103** and model optimizer **107** can be hosted by separate virtual computing instances of the cloud computing system.

[0076] Distributor **401** can be configured to provide, consistent with disclosed embodiments, an interface between the components of system **400**, and between the components of system **400** and other systems. In some embodiments, distributor **401** can be configured to implement interface **113** and a load balancer. Distributor **401** can be configured to route messages between computing resources **101** (e.g., implemented on one or more of development environment **405** and production environment **411**), dataset generator **103** (e.g., implemented on dataset generator instance **403**), and model optimizer **107** (e.g., implemented on model optimization instance **409**). The messages can include data and instructions. For example, the messages can include model generation requests and trained models provided in response to model generation requests. As an additional example, the messages can include synthetic data sets or synthetic data streams. Consistent with disclosed embodiments, distributor **401** can be implemented using one or more EC2 clusters or the like.

[0077] Data generation instance **403** can be configured to generate synthetic data, consistent with disclosed embodiments. In some embodiments, data generation instance **403** can be configured to receive actual or synthetic data from data source **417**. In various embodiments, data generation instance **403** can be configured to receive synthetic data models for generating the synthetic data. In some aspects, the synthetic data models can be received from another component of system **400**, such as data source **417**.

[0078] Development environment **405** can be configured to implement at least a portion of the functionality of computing resources **101**, consistent with disclosed embodiments. For example, development environment **405** can be configured to train data models for subsequent use by other components of system **400**. In some aspects, development instances (e.g., development instance **407**) hosted by development environment **405** can train one or more individual data models. In some aspects, development environment **405** be configured to spin up additional development instances to train additional data models, as needed. In some aspects, a development instance can implement an application framework such as TENSORBOARD, JUPYTER and the like; as well as machine learning applications like TENSORFLOW, CUDNN, KERAS, and the like. Consistent with disclosed embodiments, these application frameworks and applications can enable the specification and training of data models. In various aspects, development environment **405** can be implemented using one or more EC2 clusters or the like.

[0079] Model optimization instance **409** can be configured to manage training and provision of data models by system **400**. In some aspects, model optimization instance **409** can be configured to provide the functionality of model optimizer **107**. For example, model optimization instance **409** can be configured to provide training parameters and at least partially initialized data models to development environment **405**. This selection can be based on model performance feedback received from development environment **405**. As an additional example, model optimization instance **409** can be configured to determine whether a data model satisfies performance criteria. In some aspects, model optimization instance **409** can be configured to provide trained models and descriptive information concerning the trained models to another component of system **400**. In various aspects, model optimization instance **409** can be implemented using one or more EC2 clusters or the like.

[0080] Production environment **405** can be configured to implement at least a portion of the functionality of computing resources **101**, consistent with disclosed embodiments. For example, production environment **405** can be configured to use previously trained data models to process data received by system **400**. In some aspects, a production instance (e.g., production instance **413**) hosted by development environment **411** can be configured to process data using a previously trained data model. In some aspects, the production instance can implement an application framework such as TENSORBOARD, JUPYTER and the like; as well as machine learning applications like TENSORFLOW, CUDNN, KERAS, and the like. Consistent with disclosed embodiments, these application frameworks and applications can enable processing of data using data models. In various aspects, development environment **405** can be implemented using one or more EC2 clusters or the like.

[0081] A component of system **400** (e.g., model optimization instance **409**) can determine the data model and data source for a production instance according to the purpose of the data processing. For example, system **400** can configure a production instance to produce synthetic data for consumption by other systems. In this example, the production instance can then provide synthetic data for testing another application. As a further example, system **400** can configure a production instance to generate outputs using actual data. For example, system **400** can configure a production instance with a data model for detecting fraudulent transactions. The production instance can then receive a stream of financial transaction data and identify potentially fraudulent transactions. In some aspects, this data model may have been trained by system **400** using synthetic data created to resemble the stream of financial transaction data. System **400** can be configured to provide an indication of the potentially fraudulent transactions to another system configured to take appropriate action (e.g., reversing the transaction, contacting one or more of the parties to the transaction, or the like).

[0082] Production environment **411** can be configured to host a file system **415** for interfacing between one or more production instances and data source **417**. For example, data

source **417** can be configured to store data in file system **415**, while the one or more production instances can be configured to retrieve the stored data from file system **415** for processing. In some embodiments, file system **415** can be configured to scale as needed. In various embodiments, file system **415** can be configured to support parallel access by data source **417** and the one or more production instances. For example, file system **415** can be an instance of AMAZON ELASTIC FILE SYSTEM (EFS) or the like.

[0083] Data source **417** can be configured to provide data to other components of system **400**. In some embodiments, data source **417** can include sources of actual data, such as streams of transaction data, human resources data, web log data, web security data, web protocols data, or system logs data. System **400** can also be configured to implement model storage **109** using a database (not shown) accessible to at least one other component of system **400** (e.g., distributor **401**, dataset generation instance **403**, development environment **405**, model optimization instance **409**, or production environment **411**). In some aspects, the database can be an s3 bucket, relational database, or the like.

[0084] FIG. **5**A depicts process **500** for generating synthetic data using class-specific models, consistent with disclosed embodiments. System **100**, or a similar system, may be configured to use such synthetic data in training a data model for use in another application (e.g., a fraud detection application). Process **500** can include the steps of retrieving actual data, determining classes of sensitive portions of the data, generating synthetic data using a data model for the appropriate class, and replacing the sensitive data portions with the synthetic data portions. In some embodiments, the data model can be a generative adversarial network trained to generate synthetic data satisfying a similarity criterion, as described herein. By using class-specific models, process **500** can generate better synthetic data that more accurately models the underlying actual data than randomly generated training data that lacks the latent structures present in the actual data. Because the synthetic data more accurately models the underlying actual data, a data model trained using this improved synthetic data may perform better processing the actual data.

[0085] Process **500** can then proceed to step **501**. In step **501**, dataset generator **103** can be configured to retrieve actual data. As a non-limiting example, the actual data may have been gathered during the course of ordinary business operations, marketing operations, research operations, or the like. Dataset generator **103** can be configured to retrieve the actual data from database **105** or from another system. The actual data may have been purchased in whole or in part by an entity associated with system **100**. As would be understood from this description, the source and composition of the actual data is not intended to be limiting.

[0086] Process **500** can then proceed to step **503**. In step **503**, dataset generator **103** can be configured to determine classes of the sensitive portions of the actual data. As a non-limiting example, when the actual data is account transaction data, classes could include account numbers and merchant names. As an additional non-limiting example, when the actual data is personnel records, classes could include employee identification numbers, employee names, employee addresses, contact information, marital or beneficiary information, title and salary information, and employment actions. Consistent with disclosed embodiments, dataset generator **103** can be configured with a classifier for distinguishing different classes of sensitive information. In some embodiments, dataset generator **103** can be configured with a recurrent neural network for distinguishing different classes of sensitive information. Dataset generator **103** can be configured to apply the classifier to the actual data to determine that a sensitive portion of the training dataset belongs to the data class. For example, when the data stream includes the text string "Lorem ipsum 012-34-5678 dolor sit amet," the classifier may be configured to indicate that positions 13-23 of the text string include a potential social security number. Though described with reference to character string substitutions, the disclosed systems and methods are not so limited. As a non-limiting example, the actual data can include unstructured data (e.g., character strings, tokens, and the like) and structured data (e.g., key-value pairs, relational database files, spreadsheets, and the like).

[0087] Process **500** can then proceed to step **505**. In step **505**, dataset generator **103** can be configured to generate a synthetic portion using a class-specific model. To continue the previous example, dataset generator **103** can generate a synthetic social security number using a synthetic data model trained to generate social security numbers. In some embodiments, this class-specific synthetic data model can be trained to generate synthetic portions similar to those appearing in the actual data. For example, as social security numbers include an area number indicating geographic information and a group number indicating date-dependent information, the range of social security numbers present in an actual dataset can depend on the geographic origin and purpose of that dataset. A dataset of social security numbers for elementary school children in a particular school district may exhibit different characteristics than a dataset of social security numbers for employees of a national corporation. To continue the previous example, the social security-specific synthetic data model could generate the synthetic portion "03-74-3285."

[0088] Process **500** can then proceed to step **507**. In step **507**, dataset generator **103** can be configured to replace the sensitive portion of the actual data with the synthetic portion. To continue the previous example, dataset generator **103** could be configured to replace the characters at positions 13-23 of the text string with the values "013-74-3285," creating the synthetic text string "Lorem ipsum 013-74-3285 dolor sit amet." This text string can now be distributed without disclosing the sensitive information originally present. But this text string can still be used to train models that make valid inferences regarding the actual data, because synthetic social security numbers generated by the synthetic data model share the statistical characteristic of the actual data.

[0089] FIG. **5**B depicts a process **510** for generating synthetic data using class and subclass-specific models, consistent with disclosed embodiments. Process **510** can include the steps of retrieving actual data, determining classes of sensitive portions of the data, selecting types for synthetic data used to replace the sensitive portions of the actual data, generating synthetic data using a data model for the appropriate type and class, and replacing the sensitive data portions with the synthetic data portions. In some embodiments, the data model can be a generative adversarial network trained to generate synthetic data satisfying a similarity criterion, as described herein. This improvement addresses a problem with synthetic data generation, that a synthetic data model may fail to generate examples of

proportionately rare data subclasses. For example, when data can be classified into two distinct subclasses, with a second subclass far less prevalent in the data than a first subclass, a model of the synthetic data may generate only examples of the most common first data subclasses. The synthetic data model effectively focuses on generating the best examples of the most common data subclasses, rather than acceptable examples of all the data subclasses. Process **510** addresses this problem by expressly selecting subclasses of the synthetic data class according to a distribution model based on the actual data.

[0090] Process **510** can then proceed through step **511** and step **513**, which resemble step **501** and step **503** in process **500**. In step **511**, dataset generator **103** can be configured to receive actual data. In step **513**, dataset generator can be configured to determine classes of sensitive portions of the actual data. In a non-limiting example, dataset generator **103** can be configured to determine that a sensitive portion of the data may contain a financial service account number. Dataset generator **103** can be configured to identify this sensitive portion of the data as a financial service account number using a classifier, which may in some embodiments be a recurrent neural network (which may include LSTM units).

[0091] Process **510** can then proceed to step **515**. In step **515**, dataset generator **103** can be configured to select a subclass for generating the synthetic data. In some aspects, this selection is not governed by the subclass of the identified sensitive portion. For example, in some embodiments the classifier that identifies the class need not be sufficiently discerning to identify the subclass, relaxing the requirements on the classifier. Instead, this selection is based on a distribution model. For example, dataset generator **103** can be configured with a statistical distribution of subclasses (e.g., a univariate distribution of subclasses) for that class and can select one of the subclasses for generating the synthetic data according to the statistical distribution. To continue the previous example, individual accounts and trust accounts may both be financial service account numbers, but the values of these accounts numbers may differ between individual accounts and trust accounts. Furthermore, there may be 19 individual accounts for every 1 trust account. In this example, dataset generator **103** can be configured to select the trust account subclass 1 time in 20, and use a synthetic data model for financial service account numbers for trust accounts to generate the synthetic data. As a further example, dataset generator **103** can be configured with a recurrent neural network that estimates the next subclass based on the current and previous subclasses. For example, healthcare records can include cancer diagnosis stage as sensitive data. Most cancer diagnosis stage values may be "no cancer" and the value of "stage 1" may be rare, but when present in a patient record this value may be followed by "stage 2," etc. The recurrent neural network can be trained on the actual healthcare records to use prior and cancer diagnosis stage values when selecting the subclass. For example, when generating a synthetic healthcare record, the recurrent neural network can be configured to use the previously selected cancer diagnosis stage subclass in selecting the present cancer diagnosis stage subclass. In this manner, the synthetic healthcare record can exhibit an appropriate progression of patient health that matches the progression in the actual data.

[0092] Process **510** can then proceed to step **517**. In step **517**, which resembles step **505**, dataset generator **103** can be configured to generate synthetic data using a class and subclass specific model. To continue the previous financial service account number example, dataset generator **103** can be configured to use a synthetic data for trust account financial service account numbers to generate the synthetic financial server account number.

[0093] Process **510** can then proceed to step **519**. In step **519**, which resembles step **507**, dataset generator **103** can be configured to replace the sensitive portion of the actual data with the generated synthetic data. For example, dataset generator **103** can be configured to replace the financial service account number in the actual data with the synthetic trust account financial service account number.

[0094] FIG. **6** depicts a process **600** for training a classifier for generation of synthetic data. In some embodiments, such a classifier could be used by dataset generator **103** to classify sensitive data portions of actual data, as described above with regards to FIGS. **5**A and **5**B. Process **600** can include the steps of receiving data sequences, receiving content sequences, generating training sequences, generating label sequences, and training a classifier using the training sequences and the label sequences. By using known data sequences and content sequences unlikely to contain sensitive data, process **600** can be used to automatically generate a corpus of labeled training data. Process **600** can be performed by a component of system **100**, such as dataset generator **103** or model optimizer **107**.

[0095] Process **600** can then proceed to step **601**. In step **601**, system **100** can receive training data sequences. The training data sequences can be received from a dataset. The dataset providing the training data sequences can be a component of system **100** (e.g., database **105**) or a component of another system. The data sequences can include multiple classes of sensitive data. As a non-limiting example, the data sequences can include account numbers, social security numbers, and full names.

[0096] Process **600** can then proceed to step **603**. In step **603**, system **100** can receive context sequences. The context sequences can be received from a dataset. The dataset providing the context sequences can be a component of system **100** (e.g., database **105**) or a component of another system. In various embodiments, the context sequences can be drawn from a corpus of pre-existing data, such as an open-source text dataset (e.g., Yelp Open Dataset or the like). In some aspects, the context sequences can be snippets of this pre-existing data, such as a sentence or paragraph of the pre-existing data.

[0097] Process **600** can then proceed to step **605**. In step **605**, system **100** can generate training sequences. In some embodiments, system **100** can be configured to generate a training sequence by inserting a data sequence into a context sequence. The data sequence can be inserted into the context sequence without replacement of elements of the context sequence or with replacement of elements of the context sequence. The data sequence can be inserted into the context sequence between elements (e.g., at a whitespace character, tab, semicolon, html closing tag, or other semantic breakpoint) or without regard to the semantics of the context sequence. For example, when the context sequence is "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod" and the data sequence is "013-74-3285," the training sequence can be "Lorem ipsum dolor sit amet, 013-74-3285 consectetur adipiscing elit, sed do eiusmod," "Lorem ipsum dolor sit amet, 013-74-3285 adipiscing elit,

sed do eiusmod," or "Lorem ipsum dolor sit amet, conse013-74-3285ctetur adipiscing elit, sed do eiusmod." In some embodiments, a training sequence can include multiple data sequences.

[0098] After step **601** and step **603**, process **600** can proceed to step **607**. In step **607**, system **100** can generate a label sequence. In some aspects, the label sequence can indicate a position of the inserted data sequence in the training sequence. In various aspects, the label sequence can indicate the class of the data sequence. As a non-limiting example, when the training sequence is "dolor sit amet, 013-74-3285 consectetur adipiscing," the label sequence can be "00000000000000001111111111100000000000000000000," where the value "0" indicates that a character is not part of a sensitive data portion and the value "1" indicates that a character is part of the social security number. A different class or subclass of data sequence could include a different value specific to that class or subclass. Because system **100** creates the training sequences, system **100** can automatically create accurate labels for the training sequences.

[0099] Process **600** can then proceed to step **609**. In step **609**, system **100** can be configured to use the training sequences and the label sequences to train a classifier. In some aspects, the label sequences can provide a "ground truth" for training a classifier using supervised learning. In some embodiments, the classifier can be a recurrent neural network (which may include LSTM units). The recurrent neural network can be configured to predict whether a character of a training sequence is part of a sensitive data portion. This prediction can be checked against the label sequence to generate an update to the weights and offsets of the recurrent neural network. This update can then be propagated through the recurrent neural network, according to methods described in "Training Recurrent Neural Networks," 2013, by Ilya Sutskever, which is incorporated herein by reference in its entirety.

[0100] FIG. **7** depicts a process **700** for training a classifier for generation of synthetic data, consistent with disclosed embodiments. According to process **700**, a data sequence **701** can include preceding samples **703**, current sample **705**, and subsequent samples **707**. In some embodiments, data sequence **701** can be a subset of a training sequence, as described above with regard to FIG. **6**. Data sequence **701** may be applied to recurrent neural network **709**. In some embodiments, neural network **709** can be configured to estimate whether current sample **705** is part of a sensitive data portion of data sequence **701** based on the values of preceding samples **703**, current sample **705**, and subsequent samples **707**. In some embodiments, preceding samples **703** can include between 1 and 100 samples, for example between 25 and 75 samples. In various embodiments, subsequent samples **707** can include between 1 and 100 samples, for example between 25 and 75 samples. In some embodiments, the preceding samples **703** and the subsequent samples **707** can be paired and provided to recurrent neural network **709** together. For example, in a first iteration, the first sample of preceding samples **703** and the last sample of subsequent samples **707** can be provided to recurrent neural network **709**. In the next iteration, the second sample of preceding samples **703** and the second-to-last sample of subsequent samples **707** can be provided to recurrent neural network **709**. System **100** can continue to provide samples to recurrent neural network **709** until all of preceding samples **703** and subsequent samples **707** have been input to recurrent neural network **709**. System **100** can then provide current sample **705** to recurrent neural network **709**. The output of recurrent neural network **709** after the input of current sample **705** can be estimated label **711**. Estimated label **711** can be the inferred class or subclass of current sample **705**, given data sequence **701** as input. In some embodiments, estimated label **711** can be compared to actual label **713** to calculate a loss function. Actual label **713** can correspond to data sequence **701**. For example, when data sequence **701** is a subset of a training sequence, actual label **713** can be an element of the label sequence corresponding to the training sequence. In some embodiments, actual label **713** can occupy the same position in the label sequence as occupied by current sample **705** in the training sequence. Consistent with disclosed embodiments, system **100** can be configured to update recurrent neural network **709** using loss function **715** based on a result of the comparison.

[0101] FIG. **8** depicts a process **800** for training a generative adversarial network using a normalized reference dataset. In some embodiments, the generative adversarial network can be used by system **100** (e.g., by dataset generator **103**) to generate synthetic data (e.g., as described above with regards to FIGS. **2**, **3**, **5**A and **5**B). The generative adversarial network can include a generator network and a discriminator network. The generator network can be configured to learn a mapping from a sample space (e.g., a random number or vector) to a data space (e.g. the values of the sensitive data). The discriminator can be configured to determine, when presented with either an actual data sample or a sample of synthetic data generated by the generator network, whether the sample was generated by the generator network or was a sample of actual data. As training progresses, the generator can improve at generating the synthetic data and the discriminator can improve at determining whether a sample is actual or synthetic data. In this manner, a generator can be automatically trained to generate synthetic data similar to the actual data. However, a generative adversarial network can be limited by the actual data. For example, an unmodified generative adversarial network may be unsuitable for use with categorical data or data including missing values, not-a-numbers, or the like. For example, the generative adversarial network may not know how to interpret such data. Disclosed embodiments address this technical problem by at least one of normalizing categorical data or replacing missing values with supra-normal values.

[0102] Process **800** can then proceed to step **801**. In step **801**, system **100** (e.g., dataset generator **103**) can retrieve a reference dataset from a database (e.g., database **105**). The reference dataset can include categorical data. For example, the reference dataset can include spreadsheets or relational databases with categorical-valued data columns. As a further example, the reference dataset can include missing values, not-a-number values, or the like.

[0103] Process **800** can then proceed to step **803**. In step **803**, system **100** (e.g., dataset generator **103**) can generate a normalized training dataset by normalizing the reference dataset. For example, system **100** can be configured to normalize categorical data contained in the reference dataset. In some embodiments, system **100** can be configured to normalize the categorical data by converting this data to numerical values. The numerical values can lie within a predetermined range. In some embodiments, the predetermined range can be zero to one. For example, given a

column of categorical data including the days of the week, system **100** can be configured to map these days to values between zero and one. In some embodiments, system **100** can be configured to normalize numerical data in the reference dataset as well, mapping the values of the numerical data to a predetermined range.

[0104] Process **800** can then proceed to step **805**. In step **805**, system **100** (e.g., dataset generator **103**) can generate the normalized training dataset by converting special values to values outside the predetermined range. For example, system **100** can be configured to assign missing values a first numerical value outside the predetermined range. As an additional example, system **100** can be configured to assign not-a-number values to a second numerical value outside the predetermined range. In some embodiments, the first value and the second value can differ. For example, system **100** can be configured to map the categorical values and the numerical values to the range of zero to one. In some embodiments, system **100** can then map missing values to the numerical value 1.5. In various embodiments, system **100** can then map not-a-number values to the numerical value of −0.5. In this manner system **100** can preserve information about the actual data while enabling training of the generative adversarial network.

[0105] Process **800** can then proceed to step **807**. In step **807**, system **100** (e.g., dataset generator **103**) can train the generative network using the normalized dataset, consistent with disclosed embodiments.

[0106] FIG. **9** depicts a process **900** for training a generative adversarial network using a loss function configured to ensure a predetermined degree of similarity, consistent with disclosed embodiments. System **100** can be configured to use process **900** to generate synthetic data that is similar, but not too similar to the actual data, as the actual data can include sensitive personal information. For example, when the actual data includes social security numbers or account numbers, the synthetic data would preferably not simply recreate these numbers. Instead, system **100** would preferably create synthetic data that resembles the actual data, as described below, while reducing the likelihood of overlapping values. To address this technical problem, system **100** can be configured to determine a similarity metric value between the synthetic dataset and the normalized reference dataset, consistent with disclosed embodiments. System **100** can be configured to use the similarity metric value to update a loss function for training the generative adversarial network. In this manner, system **100** can be configured to determine a synthetic dataset differing in value from the normalized reference dataset at least a predetermined amount according to the similarity metric.

[0107] While described below with regard to training a synthetic data model, dataset generator **103** can be configured to use such trained synthetic data models to generate synthetic data (e.g., as described above with regards to FIGS. **2** and **3**). For example, development instances (e.g., development instance **407**) and production instances (e.g., production instance **413**) can be configured to generate data similar to a reference dataset according to the disclosed systems and methods.

[0108] Process **900** can then proceed to step **901**, which can resemble step **801**. In step **901**, system **100** (e.g., model optimizer **107**, computational resources **101**, or the like) can receive a reference dataset. In some embodiments, system **100** can be configured to receive the reference dataset from a database (e.g., database **105**). The reference dataset can include categorical and/or numerical data. For example, the reference dataset can include spreadsheet or relational database data. In some embodiments, the reference dataset can include special values, such as missing values, not-a-number values, or the like.

[0109] Process **900** can then proceed to step **903**. In step **903**, system **100** (e.g., dataset generator **103**, model optimizer **107**, computational resources **101**, or the like) can be configured to normalize the reference dataset. In some instances, system **100** can be configured to normalize the reference dataset as described above with regard to steps **803** and **805** of process **800**. For example, system **100** can be configured to normalize the categorical data and/or the numerical data in the reference dataset to a predetermined range. In some embodiments, system **100** can be configured to replace special values with numerical values outside the predetermined range.

[0110] Process **900** can then proceed to step **905**. In step **905**, system **100** (e.g., model optimizer **107**, computational resources **101**, or the like) can generate a synthetic training dataset using the generative network. For example, system **100** can apply one or more random samples to the generative network to generate one or more synthetic data items. In some instances, system **100** can be configured to generate between 200 and 400,000 data items, or preferably between 20,000 and 40,000 data items.

[0111] Process **900** can then proceed to step **907**. In step **907**, system **100** (e.g., model optimizer **107**, computational resources **101**, or the like) can determine a similarity metric value using the normalized reference dataset and the synthetic training dataset. System **100** can be configured to generate the similarity metric value according to a similarity metric. In some aspects, the similarity metric value can include at least one of a statistical correlation score (e.g., a score dependent on the covariances or univariate distributions of the synthetic data and the normalized reference dataset), a data similarity score (e.g., a score dependent on a number of matching or similar elements in the synthetic dataset and normalized reference dataset), or data quality score (e.g., a score dependent on at least one of a number of duplicate elements in each of the synthetic dataset and normalized reference dataset, a prevalence of the most common value in each of the synthetic dataset and normalized reference dataset, a maximum difference of rare values in each of the synthetic dataset and normalized reference dataset, the differences in schema between the synthetic dataset and normalized reference dataset, or the like). System **100** can be configured to calculate these scores using the synthetic dataset and a reference dataset.

[0112] In some aspects, the similarity metric can depend on a covariance of the synthetic dataset and a covariance of the normalized reference dataset. For example, in some embodiments, system **100** can be configured to generate a difference matrix using a covariance matrix of the normalized reference dataset and a covariance matrix of the synthetic dataset. As a further example, the difference matrix can be the difference between the covariance matrix of the normalized reference dataset and the covariance matrix of the synthetic dataset. The similarity metric can depend on the difference matrix. In some aspects, the similarity metric can depend on the summation of the squared values of the difference matrix. This summation can be normalized, for example by the square root of the product of the number of

rows and number of columns of the covariance matrix for the normalized reference dataset.

[0113] In some embodiments, the similarity metric can depend on a univariate value distribution of an element of the synthetic dataset and a univariate value distribution of an element of the normalized reference dataset. For example, for corresponding elements of the synthetic dataset and the normalized reference dataset, system **100** can be configured to generate histograms having the same bins. For each bin, system **100** can be configured to determine a difference between the value of the bin for the synthetic data histogram and the value of the bin for the normalized reference dataset histogram. In some embodiments, the values of the bins can be normalized by the total number of datapoints in the histograms. For each of the corresponding elements, system **100** can be configured to determine a value (e.g., a maximum difference, an average difference, a Euclidean distance, or the like) of these differences. In some embodiments, the similarity metric can depend on a function of this value (e.g., a maximum, average, or the like) across the common elements. For example, the normalized reference dataset can include multiple columns of data. The synthetic dataset can include corresponding columns of data. The normalized reference dataset and the synthetic dataset can include the same number of rows. System **100** can be configured to generate histograms for each column of data for each of the normalized reference dataset and the synthetic dataset. For each bin, system **100** can determine the difference between the count of datapoints in the normalized reference dataset histogram and the synthetic dataset histogram. System **100** can determine the value for this column to be the maximum of the differences for each bin. System **100** can determine the value for the similarity metric to be the average of the values for the columns. As would be appreciated by one of skill in the art, this example is not intended to be limiting.

[0114] In various embodiments, the similarity metric can depend on a number of elements of the synthetic dataset that match elements of the reference dataset. In some embodiments, the matching can be an exact match, with the value of an element in the synthetic dataset matching the value of an element in the normalized reference dataset. As a non-limiting example, when the normalized reference dataset includes a spreadsheet having rows and columns, and the synthetic dataset includes a spreadsheet having rows and corresponding columns, the similarity metric can depend on the number of rows of the synthetic dataset that have the same values as rows of the normalized reference dataset. In some embodiments, the normalized reference dataset and synthetic dataset can have duplicate rows removed prior to performing this comparison. System **100** can be configured to merge the non-duplicate normalized reference dataset and non-duplicate synthetic dataset by all columns. In this non-limiting example, the size of the resulting dataset will be the number of exactly matching rows. In some embodiments, system **100** can be configured to disregard columns that appear in one dataset but not the other when performing this comparison.

[0115] In various embodiments, the similarity metric can depend on a number of elements of the synthetic dataset that are similar to elements of the normalized reference dataset. System **100** can be configured to calculate similarity between an element of the synthetic dataset and an element of the normalized reference dataset according to distance measure. In some embodiments, the distance measure can depend on a Euclidean distance between the elements. For example, when the synthetic dataset and the normalized reference dataset include rows and columns, the distance measure can depend on a Euclidean distance between a row of the synthetic dataset and a row of the normalized reference dataset. In various embodiments, when comparing a synthetic dataset to an actual dataset including categorical data (e.g., a reference dataset that has not been normalized), the distance measure can depend on a Euclidean distance between numerical row elements and a Hamming distance between non-numerical row elements. The Hamming distance can depend on a count of non-numerical elements differing between the row of the synthetic dataset and the row of the actual dataset. In some embodiments, the distance measure can be a weighted average of the Euclidean distance and the Hamming distance. In some embodiments, system **100** can be configured to disregard columns that appear in one dataset but not the other when performing this comparison. In various embodiments, system **100** can be configured to remove duplicate entries from the synthetic dataset and the normalized reference dataset before performing the comparison.

[0116] In some embodiments, system **100** can be configured to calculate a distance measure between each row of the synthetic dataset (or a subset of the rows of the synthetic dataset) and each row of the normalized reference dataset (or a subset of the rows of the normalized reference dataset). System **100** can then determine the minimum distance value for each row of the synthetic dataset across all rows of the normalized reference dataset. In some embodiments, the similarity metric can depend on a function of the minimum distance values for all rows of the synthetic dataset (e.g., a maximum value, an average value, or the like).

[0117] In some embodiments, the similarity metric can depend on a frequency of duplicate elements in the synthetic dataset and the normalized reference dataset. In some aspects, system **100** can be configured to determine the number of duplicate elements in each of the synthetic dataset and the normalized reference dataset. In various aspects, system **100** can be configured to determine the proportion of each dataset represented by at least some of the elements in each dataset. For example, system **100** can be configured to determine the proportion of the synthetic dataset having a particular value. In some aspects, this value may be the most frequent value in the synthetic dataset. System **100** can be configured to similarly determine the proportion of the normalized reference dataset having a particular value (e.g., the most frequent value in the normalized reference dataset).

[0118] In some embodiments, the similarity metric can depend on a relative prevalence of rare values in the synthetic and normalized reference dataset. In some aspects, such rare values can be those present in a dataset with frequencies less than a predetermined threshold. In some embodiments, the predetermined threshold can be a value less than 20%, for example 10%. System **100** can be configured to determine a prevalence of rare values in the synthetic and normalized reference dataset. For example, system **100** can be configured to determine counts of the rare values in a dataset and the total number of elements in the dataset. System **100** can then determine ratios of the counts of the rare values to the total number of elements in the datasets.

[0119] In some embodiments, the similarity metric can depend on differences in the ratios between the synthetic

dataset and the normalized reference dataset. As a non-limiting example, an exemplary dataset can be an access log for patient medical records that tracks the job title of the employee accessing a patient medical record. The job title "Administrator" may be a rare value of job title and appear in 3% of the log entries. System **100** can be configured to generate synthetic log data based on the actual dataset, but the job title "Administrator" may not appear in the synthetic log data. The similarity metric can depend on difference between the actual dataset prevalence (3%) and the synthetic log data prevalence (0%). As an alternative example, the job title "Administrator" may be overrepresented in the synthetic log data, appearing in 15% of the of the log entries (and therefore not a rare value in the synthetic log data when the predetermined threshold is 10%). In this example, the similarity metric can depend on difference between the actual dataset prevalence (3%) and the synthetic log data prevalence (15%).

[0120] In various embodiments, the similarity metric can depend on a function of the differences in the ratios between the synthetic dataset and the normalized reference dataset. For example, the actual dataset may include 10 rare values with a prevalence under 10% of the dataset. The difference between the prevalence of these 10 rare values in the actual dataset and the normalized reference dataset can range from −5% to 4%. In some embodiments, the similarity metric can depend on the greatest magnitude difference (e.g., the similarity metric could depend on the value −5% as the greatest magnitude difference). In various embodiments, the similarity metric can depend on the average of the magnitude differences, the Euclidean norm of the ratio differences, or the like.

[0121] In various embodiments, the similarity metric can depend on a difference in schemas between the synthetic dataset and the normalized reference dataset. For example, when the synthetic dataset includes spreadsheet data, system **100** can be configured to determine a number of mismatched columns between the synthetic and normalized reference datasets, a number of mismatched column types between the synthetic and normalized reference datasets, a number of mismatched column categories between the synthetic and normalized reference datasets, and number of mismatched numeric ranges between the synthetic and normalized reference datasets. The value of the similarity metric can depend on the number of at least one of the mismatched columns, mismatched column types, mismatched column categories, or mismatched numeric ranges.

[0122] In some embodiments, the similarity metric can depend on one or more of the above criteria. For example, the similarity metric can depend on one or more of (1) a covariance of the output data and a covariance of the normalized reference dataset, (2) a univariate value distribution of an element of the synthetic dataset, (3) a univariate value distribution of an element of the normalized reference dataset, (4) a number of elements of the synthetic dataset that match elements of the reference dataset, (5) a number of elements of the synthetic dataset that are similar to elements of the normalized reference dataset, (6) a distance measure between each row of the synthetic dataset (or a subset of the rows of the synthetic dataset) and each row of the normalized reference dataset (or a subset of the rows of the normalized reference dataset), (7) a frequency of duplicate elements in the synthetic dataset and the normalized reference dataset, (8) a relative prevalence of rare values in the synthetic and normalized reference dataset, and (9) differences in the ratios between the synthetic dataset and the normalized reference dataset.

[0123] System **100** can compare a synthetic dataset to a normalized reference dataset, a synthetic dataset to an actual (unnormalized) dataset, or to compare two datasets according to a similarity metric consistent with disclosed embodiments. For example, in some embodiments, model optimizer **107** can be configured to perform such comparisons. In various embodiments, model storage **105** can be configured to store similarity metric information (e.g., similarity values, indications of comparison datasets, and the like) together with a synthetic dataset.

[0124] Process **900** can then proceed to step **909**. In step **909**, system **100** (e.g., model optimizer **107**, computational resources **101**, or the like) can train the generative adversarial network using the similarity metric value. In some embodiments, system **100** can be configured to determine that the synthetic dataset satisfies a similarity criterion. The similarity criterion can concern at least one of the similarity metrics described above. For example, the similarity criterion can concern at least one of a statistical correlation score between the synthetic dataset and the normalized reference dataset, a data similarity score between the synthetic dataset and the reference dataset, or a data quality score for the synthetic dataset.

[0125] In some embodiments, synthetic data satisfying the similarity criterion can be too similar to the reference dataset. System **100** can be configured to update a loss function for training the generative adversarial network to decrease the similarity between the reference dataset and synthetic datasets generated by the generative adversarial network when the similarity criterion is satisfied. In particular, the loss function of the generative adversarial network can be configured to penalize generation of synthetic data that is too similar to the normalized reference dataset, up to a certain threshold. To that end, a penalty term can be added to the loss function of the generative adversarial network. This term can penalize the calculated loss if the dissimilarity between the synthetic data and the actual data goes below a certain threshold. In some aspects, this penalty term can thereby ensure that the value of the similarity metric exceeds some similarity threshold, or remains near the similarity threshold (e.g., the value of the similarity metric may exceed 90% of the value of the similarity threshold). In this non-limiting example, decreasing values of the similarity metric can indicate increasing similarity. System **100** can then update the loss function such that the likelihood of generating synthetic data like the current synthetic data is reduced. In this manner, system **100** can train the generative adversarial network using a loss function that penalizes generation of data differing from the reference dataset by less than the predetermined amount.

[0126] FIG. **10** depicts a process **1000** for supplementing or transforming datasets using code-space operations, consistent with disclosed embodiments. Process **1000** can include the steps of generating encoder and decoder models that map between a code space and a sample space, identifying representative points in code space, generating a difference vector in code space, and generating extreme points or transforming a dataset using the difference vector. In this manner, process **1000** can support model validation and simulation of conditions differing from those present during generation of a training dataset. For example, while

existing systems and methods may train models using datasets representative of typical operating conditions, process **1000** can support model validation by inferring datapoints that occur infrequently or outside typical operating conditions. As an additional example, a training data include operations and interactions typical of a first user population. Process **1000** can support simulation of operations and interactions typical of a second user population that differs from the first user population. To continue this example, a young user population may interact with a system. Process **1000** can support generation of a synthetic training dataset representative of an older user population interacting with the system. This synthetic training dataset can be used to simulate performance of the system with an older user population, before developing that userbase.

[0127] After starting, process **1000** can proceed to step **1001**. In step **1001**, system **1001** can generate an encoder model and a decoder model. Consistent with disclosed embodiments, system **100** can be configured to generate an encoder model and decoder model using an adversarially learned inference model, as disclosed in "Adversarially Learned Inference" by Vincent Dumoulin, et al. According to the adversarially learned inference model, an encoder maps from a sample space to a code space and a decoder maps from a code space to a sample space. The encoder and decoder are trained by selecting either a code and generating a sample using the decoder or by selecting a sample and generating a code using the encoder. The resulting pairs of code and sample are provided to a discriminator model, which is trained to determine whether the pairs of code and sample came from the encoder or decoder. The encoder and decoder can be updated based on whether the discriminator correctly determined the origin of the samples. Thus, the encoder and decoder can be trained to fool the discriminator. When appropriately trained, the joint distribution of code and sample for the encoder and decoder match. As would be appreciated by one of skill in the art, other techniques of generating a mapping from a code space to a sample space may also be used. For example, a generative adversarial network can be used to learn a mapping from the code space to the sample space.

[0128] Process **1000** can then proceed to step **1003**. In step **1003**, system **100** can identify representative points in the code space. System **100** can identify representative points in the code space by identifying points in the sample space, mapping the identified points into code space, and determining the representative points based on the mapped points, consistent with disclosed embodiments. In some embodiments, the identified points in the sample space can be elements of a dataset (e.g., an actual dataset or a synthetic dataset generated using an actual dataset).

[0129] System **100** can identify points in the sample space based on sample space characteristics. For example, when the sample space includes financial account information, system **100** can be configured to identify one or more first accounts belonging to users in their 20s and one or more second accounts belonging to users in their 40s.

[0130] Consistent with disclosed embodiments, identifying representative points in the code space can include a step of mapping the one or more first points in the sample space and the one or more second points in the sample space to corresponding points in the code space. In some embodiments, the one or more first points and one or more second points can be part of a dataset. For example, the one or more first points and one or more second points can be part of an actual dataset or a synthetic dataset generated using an actual dataset.

[0131] System **100** can be configured to select first and second representative points in the code space based on the mapped one or more first points and the mapped one or more second points. As shown in FIG. **11**A, when the one or more first points include a single point, the mapping of this single point to the code space (e.g., point **1101**) can be a first representative point in code space **1100**. Likewise, when the one or more second points include a single point, the mapping of this single point to the code space (e.g., point **1103**) can be a second representative point in code space **1100**.

[0132] As shown in FIG. **11**B, when the one or more first points include multiple points, system **100** can be configured to determine a first representative point in code space **1110**. In some embodiments, system **100** can be configured to determine the first representative point based on the locations of the mapped one or more first points in the code space. In some embodiments, the first representative point can be a centroid or a medoid of the mapped one or more first points. Likewise, system **100** can be configured to determine the second representative point based on the locations of the mapped one or more second points in the code space. In some embodiments, the second representative point can be a centroid or a medoid of the mapped one or more second points. For example, system **100** can be configured to identify point **1113** as the first representative point based on the locations of mapped points **1111***a* and **1111***b*. Likewise, system **100** can be configured to identify point **1117** as the second representative point based on the locations of mapped points **1115***a* and **1115***b*.

[0133] In some embodiments, the code space can include a subset of R^n. System **100** can be configured to map a dataset to the code space using the encoder. System **100** can then identify the coordinates of the points with respect to a basis vector in R^n (e.g., one of the vectors of the identity matrix). System **100** can be configured to identify a first point with a minimum coordinate value with respect to the basis vector and a second point with a maximum coordinate value with respect to the basis vector. System **100** can be configured to identify these points as the first and second representative points. For example, taking the identity matrix as the basis, system **100** can be configured to select as the first point the point with the lowest value of the first element of the vector. To continue this example, system **100** can be configured to select as the second point the point with the highest value of the first element of the vector. In some embodiments, system **100** can be configured to repeat process **1000** for each vector in the basis.

[0134] Process **1000** can then proceed to step **1005**. In step **1005**, system **100** can determine a difference vector connecting the first representative point and the second representative point. For example, as shown in FIG. **11**A, system **100** can be configured to determine a vector **1105** from first representative point **1101** to second representative point **1103**. Likewise, as shown in FIG. **11**B, system **100** can be configured to determine a vector **1119** from first representative point **1113** to second representative point **1117**.

[0135] Process **1000** can then proceed to step **1007**. In step **1007**, as depicted in FIG. **12**A, system **100** can generate extreme codes. Consistent with disclosed embodiments, system **100** can be configured to generate extreme codes by

sampling the code space (e.g., code space **1200**) along an extension (e.g., extension **1201**) of the vector connecting the first representative point and the second representative point (e.g., vector **1105**). In this manner, system **100** can generate a code extreme with respect to the first representative point and the second representative point (e.g. extreme point **1203**).

[0136] Process **1000** can then proceed to step **1009**. In step **1009**, as depicted in FIG. **12**A, system **100** can generate extreme samples. Consistent with disclosed embodiments, system **100** can be configured to generate extreme samples by converting the extreme code into the sample space using the decoder trained in step **1001**. For example, system **100** can be configured to convert extreme point **1203** into a corresponding datapoint in the sample space.

[0137] Process **1000** can then proceed to step **1011**. In step **1011**, as depicted in FIG. **12**B, system **100** can translate a dataset using the difference vector determined in step **1005** (e.g., difference vector **1105**). In some aspects, system **100** can be configured to convert the dataset from sample space to code space using the encoder trained in step **1001**. System **100** can be configured to then translate the elements of the dataset in code space using the difference vector. In some aspects, system **100** can be configured to translate the elements of the dataset using the vector and a scaling factor. In some aspects, the scaling factor can be less than one. In various aspects, the scaling factor can be greater than or equal to one. For example, as shown in FIG. **12**B, the elements of the dataset can be translated in code space **1210** by the product of the difference vector and the scaling factor (e.g., original point **1211** can be translated by translation **1212** to translated point **1213**).

[0138] Process **1000** can then proceed to step **1013**. In step **1013**, as depicted in FIG. **12**B, system **100** can generate a translated dataset. Consistent with disclosed embodiments, system **100** can be configured to generate the translated dataset by converting the translated points into the sample space using the decoder trained in step **1001**. For example, system **100** can be configured to convert extreme point translated point **1213** into a corresponding datapoint in the sample space.

[0139] FIG. **13** depicts an exemplary cloud computing system **1300** for generating a synthetic data stream that tracks a reference data stream. The flow rate of the synthetic data can resemble the flow rate of the reference data stream, as system **1300** can generate synthetic data in response to receiving reference data stream data. System **1300** can include a streaming data source **1301**, model optimizer **1303**, computing resource **1304**, model storage **1305**, dataset generator **1307**, and synthetic data source **1309**. System **1300** can be configured to generate a new synthetic data model using actual data received from streaming data source **1301**. Streaming data source **1301**, model optimizer **1303**, computing resources **1304**, and model storage **1305** can interact to generate the new synthetic data model, consistent with disclosed embodiments. In some embodiments, system **1300** can be configured to generate the new synthetic data model while also generating synthetic data using a current synthetic data model.

[0140] Streaming data source **1301** can be configured to retrieve new data elements from a database, a file, a data-source, a topic in a data streaming platform (e.g., IBM STREAMS), a topic in a distributed messaging system (e.g., APACHE KAFKA), or the like. In some aspects, streaming data source **1301** can be configured to retrieve new elements in response to a request from model optimizer **1303**. In some aspects, streaming data source **1301** can be configured to retrieve new data elements in real-time. For example, streaming data source **1301** can be configured to retrieve log data, as that log data is created. In various aspects, streaming data source **1301** can be configured to retrieve batches of new data. For example, streaming data source **1301** can be configured to periodically retrieve all log data created within a certain period (e.g., a five-minute interval). In some embodiments, the data can be application logs. The application logs can include event information, such as debugging information, transaction information, user information, user action information, audit information, service information, operation tracking information, process monitoring information, or the like. In some embodiments, the data can be JSON data (e.g., JSON application logs).

[0141] System **1300** can be configured to generate a new synthetic data model, consistent with disclosed embodiments. Model optimizer **1303** can be configured to provision computing resources **1304** with a data model, consistent with disclosed embodiments. In some aspects, computing resources **1304** can resemble computing resources **101**, described above with regard to FIG. **1**. For example, computing resources **1304** can provide similar functionality and can be similarly implemented. The data model can be a synthetic data model. The data model can be a current data model configured to generate data similar to recently received data in the reference data stream. The data model can be received from model storage **1305**. For example, model optimizer **1307** can be configured to provide instructions to computing resources **1304** to retrieve a current data model of the reference data stream from model storage **1305**. In some embodiments, the synthetic data model can include a recurrent neural network, a kernel density estimator, or a generative adversarial network.

[0142] Computing resources **1304** can be configured to train the new synthetic data model using reference data stream data. In some embodiments, system **1300** (e.g., computing resources **1304** or model optimizer **1303**) can be configured to include reference data stream data into the training data as it is received from streaming data source **1301**. The training data can therefore reflect the current characteristics of the reference data stream (e.g., the current values, current schema, current statistical properties, and the like). In some aspects, system **1300** (e.g., computing resources **1304** or model optimizer **1303**) can be configured to store reference data stream data received from streaming data source **1301** for subsequent use as training data. In some embodiments, computing resources **1304** may have received the stored reference data stream data prior to beginning training of the new synthetic data model. As an additional example, computing resources **1304** (or another component of system **1300**) can be configured to gather data from streaming data source **1301** during a first time-interval (e.g., the prior repeat) and use this gathered data to train a new synthetic model in a subsequent time-interval (e.g., the current repeat). In various embodiments, computing resources **1304** can be configured to use the stored reference data stream data for training the new synthetic data model. In various embodiments, the training data can include both newly-received and stored data. When the synthetic data model is a Generative Adversarial Network, computing resources **1304** can be configured to train the new synthetic

data model, in some embodiments, as described above with regard to FIGS. **8** and **9**. Alternatively, computing resources **1304** can be configured to train the new synthetic data model according to know methods.

[0143] Model optimizer **1303** can be configured to evaluate performance criteria of a newly created synthetic data model. In some embodiments, the performance criteria can include a similarity metric (e.g., a statistical correlation score, data similarity score, or data quality score, as described herein). For example, model optimizer **1303** can be configured to compare the covariances or univariate distributions of a synthetic dataset generated by the new synthetic data model and a reference data stream dataset. Likewise, model optimizer **1303** can be configured to evaluate the number of matching or similar elements in the synthetic dataset and reference data stream dataset. Furthermore, model optimizer **1303** can be configured to evaluate a number of duplicate elements in each of the synthetic dataset and reference data stream dataset, a prevalence of the most common value in synthetic dataset and reference data stream dataset, a maximum difference of rare values in each of the synthetic dataset and reference data stream dataset, differences in schema between the synthetic dataset and reference data stream dataset, and the like.

[0144] In various embodiments, the performance criteria can include prediction metrics. The prediction metrics can enable a user to determine whether data models perform similarly for both synthetic and actual data. The prediction metrics can include a prediction accuracy check, a prediction accuracy cross check, a regression check, a regression cross check, and a principal component analysis check. In some aspects, a prediction accuracy check can determine the accuracy of predictions made by a model (e.g., recurrent neural network, kernel density estimator, or the like) given a dataset. For example, the prediction accuracy check can receive an indication of the model, a set of data, and a set of corresponding labels. The prediction accuracy check can return an accuracy of the model in predicting the labels given the data. Similar model performance for the synthetic and original data can indicate that the synthetic data preserves the latent feature structure of the original data. In various aspects, a prediction accuracy cross check can calculate the accuracy of a predictive model that is trained on synthetic data and tested on the original data used to generate the synthetic data. In some aspects, a regression check can regress a numerical column in a dataset against other columns in the dataset, determining the predictability of the numerical column given the other columns. In some aspects, a regression error cross check can determine a regression formula for a numerical column of the synthetic data and then evaluate the predictive ability of the regression formula for the numerical column of the actual data. In various aspects, a principal component analysis check can determine a number of principal component analysis columns sufficient to capture a predetermined amount of the variance in the dataset. Similar numbers of principal component analysis columns can indicate that the synthetic data preserves the latent feature structure of the original data.

[0145] Model optimizer **1303** can be configured to store the newly created synthetic data model and metadata for the new synthetic data model in model storage **1305** based on the evaluated performance criteria, consistent with disclosed embodiments. For example, model optimizer **1303** can be configured to store the metadata and new data model in model storage when a value of a similarity metric or a prediction metric satisfies a predetermined threshold. In some embodiments, the metadata can include at least one value of a similarity metric or prediction metric. In various embodiments, the metadata can include an indication of the origin of the new synthetic data model, the data used to generate the new synthetic data model, when the new synthetic data model was generated, and the like.

[0146] System **1300** can be configured to generate synthetic data using a current data model. In some embodiments, this generation can occur while system **1300** is training a new synthetic data model. Model optimizer **1303**, model storage **1305**, dataset generator **1307**, and synthetic data source **1309** can interact to generate the synthetic data, consistent with disclosed embodiments.

[0147] Model optimizer **1303** can be configured to receive a request for a synthetic data stream from an interface (e.g., interface **113** or the like). In some aspects, model optimizer **1307** can resemble model optimizer **107**, described above with regard to FIG. **1**. For example, model optimizer **1307** can provide similar functionality and can be similarly implemented. In some aspects, requests received from the interface can indicate a reference data stream. For example, such a request can identify streaming data source **1301** and/or specify a topic or subject (e.g., a Kafka topic or the like). In response to the request, model optimizer **1307** (or another component of system **1300**) can be configured to direct generation of a synthetic data stream that tracks the reference data stream, consistent with disclosed embodiments.

[0148] Dataset generator **1307** can be configured to retrieve a current data model of the reference data stream from model storage **1305**. In some embodiments, dataset generator **1307** can resemble dataset generator **103**, described above with regard to FIG. **1**. For example, dataset generator **1307** can provide similar functionality and can be similarly implemented. Likewise, in some embodiments, model storage **1305** can resemble model storage **105**, described above with regard to FIG. **1**. For example, model storage **1305** can provide similar functionality and can be similarly implemented. In some embodiments, the current data model can resemble data received from streaming data source **1301** according to a similarity metric (e.g., a statistical correlation score, data similarity score, or data quality score, as described herein). In various embodiments, the current data model can resemble data received during a time interval extending to the present (e.g. the present hour, the present day, the present week, or the like). In various embodiments, the current data model can resemble data received during a prior time interval (e.g. the previous hour, yesterday, last week, or the like). In some embodiments, the current data model can be the most recently trained data model of the reference data stream.

[0149] Dataset generator **1307** can be configured to generate a synthetic data stream using the current data model of the reference data steam. In some embodiments, dataset generator **1307** can be configured to generate the synthetic data stream by replacing sensitive portions of the reference data steam with synthetic data, as described in FIGS. **5**A and **5**B. In various embodiments, dataset generator **1307** can be configured to generate the synthetic data stream without reference to the reference data steam data. For example, when the current data model is a recurrent neural network, dataset generator **1307** can be configured to initialize the recurrent neural network with a value string (e.g., a random

sequence of characters), predict a new value based on the value string, and then add the new value to the end of the value string. Dataset generator **1307** can then predict the next value using the updated value string that includes the new value. In some embodiments, rather than selecting the most likely new value, dataset generator **1307** can be configured to probabilistically choose a new value. As a nonlimiting example, when the existing value string is "examin" the dataset generator **1307** can be configured to select the next value as "e" with a first probability and select the next value as "a" with a second probability. As an additional example, when the current data model is a generative adversarial network or an adversarially learned inference network, dataset generator **1307** can be configured to generate the synthetic data by selecting samples from a code space, as described herein.

[0150] In some embodiments, dataset generator **1307** can be configured to generate an amount of synthetic data equal to the amount of actual data retrieved from synthetic data stream **1309**. In some aspects, the rate of synthetic data generation can match the rate of actual data generation. As a nonlimiting example, when streamlining data source **1301** retrieves a batch of 10 samples of actual data, dataset generator **1307** can be configured to generate a batch of 10 samples of synthetic data. As a further nonlimiting example, when streamlining data source **1301** retrieves a batch of actual data every 10 minutes, dataset generator **1307** can be configured to generate a batch of actual data every 10 minutes. In this manner, system **1300** can be configured to generate synthetic data similar in both content and temporal characteristics to the reference data stream data.

[0151] In various embodiments, dataset generator **1307** can be configured to provide synthetic data generated using the current data model to synthetic data source **1309**. In some embodiments, synthetic data source **1309** can be configured to provide the synthetic data received from dataset generator **1307** to a database, a file, a datasource, a topic in a data streaming platform (e.g., IBM STREAMS), a topic in a distributed messaging system (e.g., APACHE KAFKA), or the like.

[0152] As discussed above, system **1300** can be configured to track the reference data stream by repeatedly switching data models of the reference data stream. In some embodiments, dataset generator **1307** can be configured to switch between synthetic data models at a predetermined time, or upon expiration of a time interval. For example, model optimizer **1307** can be configured to switch from an old model to a current model every hour, day, week, or the like. In various embodiments, system **1300** can detect when a data schema of the reference data stream changes and switch to a current data model configured to provide synthetic data with the current schema. Consistent with disclosed embodiments, switching between synthetic data models can include dataset generator **1307** retrieving a current model from model storage **1305** and computing resources **1304** providing a new synthetic data model for storage in model storage **1305**. In some aspects, computing resources **1304** can update the current synthetic data model with the new synthetic data model and then dataset generator **1307** can retrieve the updated current synthetic data model. In various aspects, dataset generator **1307** can retrieve the current synthetic data model and then computing resources **1304** can update the current synthetic data model with the new synthetic data model. In some embodiments, model optimizer **1303** can provision computing resources **1304** with a synthetic data model for training using a new set of training data. In various embodiments, computing resources **1304** can be configured to continue updating the new synthetic data model. In this manner, a repeat of the switching process can include generation of a new synthetic data model and the replacement of a current synthetic data model by this new synthetic data model.

[0153] FIG. **14** depicts a process **1400** for generating synthetic JSON log data using the cloud computing system of FIG. **13**. Process **1400** can include the steps of retrieving reference JSON log data, training a recurrent neural network to generate synthetic data resembling the reference JSON log data, generating the synthetic JSON log data using the recurrent neural network, and validating the synthetic JSON log data. In this manner system **1300** can use process **1400** to generate synthetic JSON log data that resembles actual JSON log data.

[0154] After starting, process **1400** can proceed to step **1401**. In step **1401**, substantially as described above with regard to FIG. **13**, streaming data source **1301** can be configured to retrieve the JSON log data from a database, a file, a datasource, a topic in a distributed messaging system such Apache Kafka, or the like. The JSON log data can be retrieved in response to a request from model optimizer **1303**. The JSON log data can be retrieved in real-time, or periodically (e.g., approximately every five minutes).

[0155] Process **1400** can then proceed to step **1403**. In step **1403**, substantially as described above with regard to FIG. **13**, computing resources **1304** can be configured to train a recurrent neural network using the received data. The training of the recurrent neural network can proceed as described in "Training Recurrent Neural Networks," **2013**, by Ilya Sutskever, which is incorporated herein by reference in its entirety.

[0156] Process **1400** can then proceed to step **1405**. In step **1405**, substantially as described above with regards to FIG. **13**, dataset generator **1307** can be configured to generate synthetic JSON log data using the trained neural network. In some embodiments, dataset generator **1307** can be configured to generate the synthetic JSON log data at the same rate as actual JSON log data is received by streaming data source **1301**. For example, dataset generator **1307** can be configured to generate batches of JSON log data at regular time intervals, the number of elements in a batch dependent on the number of elements received by streaming data source **1301**. As an additional example, dataset generator **1307** can be configured to generate an element of synthetic JSON log data upon receipt of an element of actual JSON log data from streaming data source **1301**.

[0157] Process **1400** can then proceed to step **1407**. In step **1407**, dataset generator **1307** (or another component of system **1300**) can be configured to validate the synthetic data stream. For example, dataset generator **1307** can be configured to use a JSON validator (e.g., JSON SCHEMA VALIDATOR, JSONLINT, or the like) and a schema for the reference data stream to validate the synthetic data stream. In some embodiments, the schema describes key-value pairs present in the reference data stream. In some aspects, system **1300** can be configured to derive the schema from the reference data stream. In some embodiments, validating the synthetic data stream can include validating that keys present in the synthetic data stream are present in the schema. For example, when the schema includes the keys "first_

name": "type": "string" and "last_name": {"type": "string" }, system **1300** may not validate the synthetic data stream when objects in the data stream lack the "first_name" and "last_name" keys. Furthermore, in some embodiments, validating the synthetic data stream can include validating that key-value formats present in the synthetic data stream match corresponding key-value formats in the reference data stream. For example, when the schema includes the keys "first_name": "type": "string" and "last_name": {"type": "string" }, system **1300** may not validate the synthetic data stream when objects in the data stream include a numeric-valued "first_name" or "last_name".

[0158] FIG. **15** depicts a system **1500** for secure generation and insecure use of models of sensitive data. System **1500** can include a remote system **1501** and a local system **1503** that communicate using network **1505**. Remote system **1501** can be substantially similar to system **100** and be implemented, in some embodiments, as described in FIG. **4**. For example, remote system **1501** can include an interface, model optimizer, and computing resources that resemble interface **113**, model optimizer **107**, and computing resources **101**, respectively, described above with regards to FIG. **1**. For example, the interface, model optimizer, and computing resources can provide similar functionality to interface **113**, model optimizer **107**, and computing resources **101**, respectively, and can be similarly implemented. In some embodiments, remote system **1501** can be implemented using a cloud computing infrastructure. Local system **1503** can comprise a computing device, such as a smartphone, tablet, laptop, desktop, workstation, server, or the like. Network **1505** can include any combination of electronics communications networks enabling communication between components of system **1500** (similar to network **115**).

[0159] In various embodiments, remote system **1501** can be more secure than local system **1503**. For example, remote system **1501** can better protected from physical theft or computer intrusion than local system **1503**. As a non-limiting example, remote system **1501** can be implemented using AWS or a private cloud of an institution and managed at an institutional level, while the local system can be in the possession of, and managed by, an individual user. In some embodiments, remote system **1501** can be configured to comply with policies or regulations governing the storage, transmission, and disclosure of customer financial information, patient healthcare records, or similar sensitive information. In contrast, local system **1503** may not be configured to comply with such regulations.

[0160] System **1500** can be configured to perform a process of generating synthetic data. According to this process, system **1500** can train the synthetic data model on sensitive data using remote system **1501**, in compliance with regulations governing the storage, transmission, and disclosure of sensitive information. System **1500** can then transmit the synthetic data model to local system **1503**, which can be configured to use the system to generate synthetic data locally. In this manner, local system **1503** can be configured to use synthetic data resembling the sensitive information, which comply with policies or regulations governing the storage, transmission, and disclosure of such information.

[0161] According to this process, the model optimizer can receive a data model generation request from the interface. In response to the request, the model optimizer can provision computing resources with a synthetic data model. The computing resources can train the synthetic data model using a sensitive dataset (e.g., consumer financial information, patient healthcare information, or the like). The model optimizer can be configured to evaluate performance criteria of the data model (e.g., the similarity metric and prediction metrics described herein, or the like). Based on the evaluation of the performance criteria of the synthetic data model, the model optimizer can be configured to store the trained data model and metadata of the data model (e.g., values of the similarity metric and prediction metrics, of the data, the origin of the new synthetic data model, the data used to generate the new synthetic data model, when the new synthetic data model was generated, and the like). For example, the model optimizer can determine that the synthetic data model satisfied predetermined acceptability criteria based on one or more similarity and/or prediction metric value.

[0162] Local system **1503** can then retrieve the synthetic data model from remote system **1501**. In some embodiments, local system **1503** can be configured to retrieve the synthetic data model in response to a synthetic data generation request received by local system **1503**. For example, a user can interact with local system **1503** to request generation of synthetic data. In some embodiments, the synthetic data generation request can specify metadata criteria for selecting the synthetic data model. Local system **1503** can interact with remote system **1501** to select the synthetic data model based on the metadata criteria. Local system **1503** can then generate the synthetic data using the data model in response to the data generation request.

Example: Generating Cancer Data

[0163] As described above, the disclosed systems and methods can enable generation of synthetic data similar to an actual dataset (e.g., using dataset generator). The synthetic data can be generated using a data model trained on the actual dataset (e.g., as described above with regards to FIG. **9**). Such data models can include generative adversarial networks. The following code depicts the creation a synthetic dataset based on sensitive patient healthcare records using a generative adversarial network.

[0164] # The following step defines a Generative Adversarial Network data model.

[0165] model_options={'GANhDim': 498, 'GANZDim': 20, 'num_epochs': 3}

[0166] # The following step defines the delimiters present in the actual data

[0167] data_options={'delimiter': ','}

[0168] # In this example, the dataset is the publicly available University of Wisconsin Cancer dataset, a standard dataset used to benchmark machine learning prediction tasks. Given characteristics of a tumor, the task to predict whether the tumor is malignant.

[0169] data=Data(input_file_path='wisconsin_cancer_train.csv', options=data_options)

[0170] # In these steps the GAN model is trained generate data statistically similar to the actual data.

[0171] ss=SimpleSilo('GAN', model_options)

[0172] ss.train(data)

[0173] # The GAN model can now be used to generate synthetic data.

[0174] generated_data=ss.generate(num_output_samples=5000)

[0175] # The synthetic data can be saved to a file for later use in training other machine learning models for this prediction task without relying on the original data.

[0176] simplesilo.save_as_csv(generated_data, output_file_path='wisconsin_cancer_GAN.csv')

[0177] ss.save_model_into_file('cancer_data_model')

[0178] Tokenizing Sensitive Data

[0179] As described above with regard to at least FIGS. **5**A and **5**B, the disclosed systems and methods can enable identification and removal of sensitive data portions in a dataset. In this example, sensitive portions of a dataset are automatically detected and replaced with synthetic data. In this example, the dataset includes human resources records. The sensitive portions of the dataset are replaced with random values (though they could also be replaced with synthetic data that is statistically similar to the original data as described in FIGS. **5**A and **5**B). In particular, this example depicts tokenizing four columns of the dataset. In this example, the Business Unit and Active Status columns are tokenized such that all the characters in the values can be replaced by random chars of the same type while preserving format. For the column of Employee number, the first three characters of the values can be preserved but the remainder of each employee number can be tokenized. Finally, the values of the Last Day of Work column can be replaced with fully random values. All of these replacements can be consistent across the columns.

[0180] input_data=Data('hr_data.csv')

[0181] keys_for_formatted_scrub={'Business Unit': None, 'Active Status': None, 'Company': (0,3)}

[0182] keys_to_randomize=['Last Day of Work']

[0183] tokenized_data, scrub_map=input_data.tokenize(keys_for_formatted_scrub=keys_for_formatted_scrub, keys_to_randomize=keys_to_randomize) tokenized_data.save_data_into_file('hr_data_tokenized.csv')

[0184] Alternatively, the system can use the scrub map to tokenize another file in a consistent way (e.g., replace the same values with the same replacements across both files) by passing the returned scrub_map dictionary to a new application of the scrub function.

[0185] input_data_2=Data('hr_data_part2.csv')

[0186] keys_for_formatted_scrub={'Business Unit': None, 'Company': (0,3)}

[0187] keys_to_randomize=['Last Day of Work']

[0188] # to tokenize the second file, we pass the scrub_map diction to tokenize function.

[0189] tokenized_data_2, scrub_map=input_data_2.tokenize(keys_for_formatted_scrub=keys_for_formatted_scrub, keys_to_randomize=keys_to_randomize, scrub_map=scrub_map)

[0190] tokenized_data_2.save_data_into_file('hr_data_tokenized_2.csv')

[0191] In this manner, the disclosed systems and methods can be used to consistently tokenize sensitive portions of a file.

[0192] Other embodiments will be apparent to those skilled in the art from consideration of the specification and practice of the disclosed embodiments disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the disclosed embodiments being indicated by the following claims. Furthermore, although aspects of the disclosed embodiments are described as being associated with data stored in memory and other tangible computer-readable storage mediums, one skilled in the art will appreciate that these aspects can also be stored on and executed from many types of tangible computer-readable media, such as secondary storage devices, like hard disks, floppy disks, or CD-ROM, or other forms of RAM or ROM. Accordingly, the disclosed embodiments are not limited to the above-described examples, but instead are defined by the appended claims in light of their full scope of equivalents.

[0193] Moreover, while illustrative embodiments have been described herein, the scope includes any and all embodiments having equivalent elements, modifications, omissions, combinations (e.g., of aspects across various embodiments), adaptations or alterations based on the present disclosure. The elements in the claims are to be interpreted broadly based on the language employed in the claims and not limited to examples described in the present specification or during the prosecution of the application, which examples are to be construed as non-exclusive. Further, the steps of the disclosed methods can be modified in any manner, including by reordering steps or inserting or deleting steps. It is intended, therefore, that the specification and examples be considered as example only, with a true scope and spirit being indicated by the following claims and their full scope of equivalents.

1-20. (canceled)

21. A cloud computing system for generating data models, comprising:

at least one processor; and

at least one non-transitory memory storing instructions that, when executed by the at least one processor cause the cloud computing system to perform operations comprising:

normalizing a reference dataset;

receiving a similarity criterion, the similarity criterion including a predetermined difference in value between the normalized reference dataset and an output dataset of a data model;

generating a synthetic dataset for training the data model;

training the data model using the synthetic dataset, the training comprising:

generating, based on a comparison of the output dataset and the normalized reference dataset, a similarity metric of the data model,

generating a prediction metric of the data model,

evaluating the similarity metric against the similarity criterion,

evaluating the prediction metric against a prediction criterion, and

updating the data model based on the evaluations of the similarity metric and prediction metric, the updating comprising penalizing generation of synthetic data by adding a penalty term to a loss function;

repeating the training until the similarity criterion is met by the similarity metric and the prediction criterion is met by the similarity metric and the prediction metric; and

in response to the similarity criterion being met by the similarity metric and the prediction criterion being met by the prediction metric, storing the data model in a model storage.

22. The cloud computing system of claim **21**, wherein the similarity metric depends on a maximum distance or an average distance according to a distance measure between rows selected from the output dataset and at least one row selected from the reference dataset.

23. The cloud computing system of claim **21**, wherein the loss function is updatable for training the data model, the loss function is associated with a penalty term, to ensure that the value of the similarity metric exceeds a similarity threshold or remains near the similarity threshold.

24. The cloud computing system of claim **21**, wherein:

the similarity metric comprises at least one of a statistical correlation score, data similarity score, or data quality score; and

the prediction metric includes at least one of a prediction accuracy verification, a prediction accuracy cross validation, a regression verification, a regression cross validation, or a principal component analysis.

25. The cloud computing system of claim **24**, wherein the similarity metric is configured to calculate scores using the synthetic dataset and a reference dataset.

26. The cloud computing system of claim **21**, wherein the synthetic dataset differs in value from the normalized reference dataset according to a predetermined amount according to the similarity metric.

27. The cloud computing system of claim **21**, wherein:

the similarity metric depends on a covariance of the synthetic dataset and a covariance of the normalized reference dataset; and

the operations further comprise generating a difference matrix using a covariance matrix of the normalized reference dataset and a covariance matrix of the synthetic dataset.

28. The cloud computing system of claim **21**, wherein the prediction metric includes at least one of a prediction accuracy check, a prediction accuracy cross check, a regression check, a regression cross check, or a principal component analysis check.

29. The cloud computing system of claim **21**, wherein the similarity metric depends on one or more criteria, the one or more criteria comprising at least one of:

a covariance of output dataset and a covariance of the normalized reference dataset;

a univariate value distribution of an element of the synthetic dataset;

a univariate value distribution of an element of the normalized reference dataset;

a number of elements of the synthetic dataset that match elements of the reference dataset;

a number of elements of the synthetic dataset that are similar to elements of the normalized reference dataset;

a distance measure between each row of the synthetic dataset and each row of the normalized reference dataset;

a frequency of duplicate elements in the synthetic dataset and the normalized reference dataset; and

a relative prevalence of rare values in the synthetic dataset and the normalized reference dataset;

and differences in ratios between the synthetic dataset and the normalized reference dataset.

30. The cloud computing system of claim **21**, wherein the similarity criterion concerns at least one of a statistical correlation score between the synthetic data and the normalized reference dataset, a data similarity score between the synthetic dataset and the reference dataset, or a data quality score for the synthetic dataset.

31. A method for generating data models, comprising:

normalizing a reference dataset;

receiving a similarity criterion, the similarity criterion including a predetermined difference in value between the normalized reference dataset and an output dataset of the data model;

generating a synthetic dataset for training the data model;

training a data model using the synthetic dataset, the training comprising:

generating, based on a comparison of the output dataset and the normalized reference dataset, a similarity metric of the data model,

generating a prediction metric of the data model,

evaluating the similarity metric against the similarity criterion,

evaluating the prediction metric against a prediction criterion, and

updating the data model based on the evaluations of the similarity metric and prediction metric, the updating comprising penalizing generation of synthetic data by adding a penalty term to a loss function;

repeating the training until the similarity criterion is met by the similarity metric and the prediction criterion is met by the similarity metric and the prediction metric; and

in response to the similarity criterion being met by the similarity metric and the prediction criterion being met by the prediction metric, storing the data model and new metadata in a model storage.

32. The method of claim **31**, wherein the similarity criterion concerns at least one of a statistical correlation score between the synthetic data and the normalized reference dataset, a data similarity score between the synthetic dataset and the reference dataset, or a data quality score for the synthetic dataset.

33. The method of claim **31**, wherein the similarity metric depends on a maximum distance or an average distance according to a distance measure between rows selected from the output dataset and at least one row selected from the reference dataset.

34. The method of claim **31**, wherein the similarity metric comprises at least one of a statistical correlation score, data similarity score, or data quality score, and the prediction metric includes at least one of a prediction accuracy verification, a prediction accuracy cross validation, a regression verification, a regression cross validation, or a principal component analysis.

35. The method of claim **34**, wherein the similarity metric is configured to calculate scores using the synthetic dataset and a reference dataset.

36. The method of claim **31**, wherein the synthetic dataset differs in value from the normalized reference dataset according to a predetermined amount according to the similarity metric.

37. The method of claim **31**, wherein:

the similarity metric depends on a covariance of the synthetic dataset and a covariance of the normalized reference dataset; and

the operations further comprise generating a difference matrix using a covariance matrix of the normalized reference dataset and a covariance matrix of the synthetic dataset.

38. The method of claim **31**, wherein the prediction metric includes at least one of a prediction accuracy check, a prediction accuracy cross check, a regression check, a regression cross check, or a principal component analysis check.

39. The method of claim **31**, wherein the metadata includes an indication of origin of the new synthetic data model and the data used to generate the new synthetic data model.

40. A non-transitory computer-readable memory storing instructions that, when executed by at least one processor, cause the at least one processor to perform operations comprising:

normalizing a reference dataset;

receiving a similarity criterion, the similarity criterion including a predetermined difference in value between the normalized reference dataset and an output dataset of the data model;

generating a synthetic dataset for training the data model;

training a data model using the synthetic dataset, the training comprising:

generating, based on a comparison of the output dataset and the normalized reference dataset, a similarity metric of the data model,

generating a prediction metric of the data model,

evaluating the similarity metric against the similarity criterion,

evaluating the prediction metric against a prediction criterion to determine whether data models perform similarly for both the synthetic data and actual data, and

updating the data model based on the evaluations of the similarity metric and prediction metric, the updating comprising penalizing generation of synthetic data by adding a penalty term to a loss function, decreasing values of the similarity metric to indicate similarity;

repeating the training until the similarity criterion is met by the similarity metric and the prediction criterion is met by the similarity metric and the prediction metric; and

in response to the similarity criterion being met by the similarity metric and the prediction criterion being met by the prediction metric, storing the data model in a model storage, once a value of a similarity metric or prediction metric satisfies a predetermined threshold.

* * * * *