US 20220277196A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2022/0277196 A1**

Flint (43) **Pub. Date: Sep. 1, 2022**

(54) **METHOD AND SYSTEM FOR SECURELY STORING DATA FOR USE WITH ARTIFICIAL NEURAL NETWORKS**

(71) Applicant: **Alexander Calhoun Flint**, Birchgrove (AU)

(72) Inventor: **Alexander Calhoun Flint**, Birchgrove (AU)

(21) Appl. No.: **17/652,743**

(22) Filed: **Feb. 28, 2022**

Related U.S. Application Data

(60) Provisional application No. 63/155,210, filed on Mar. 1, 2021.

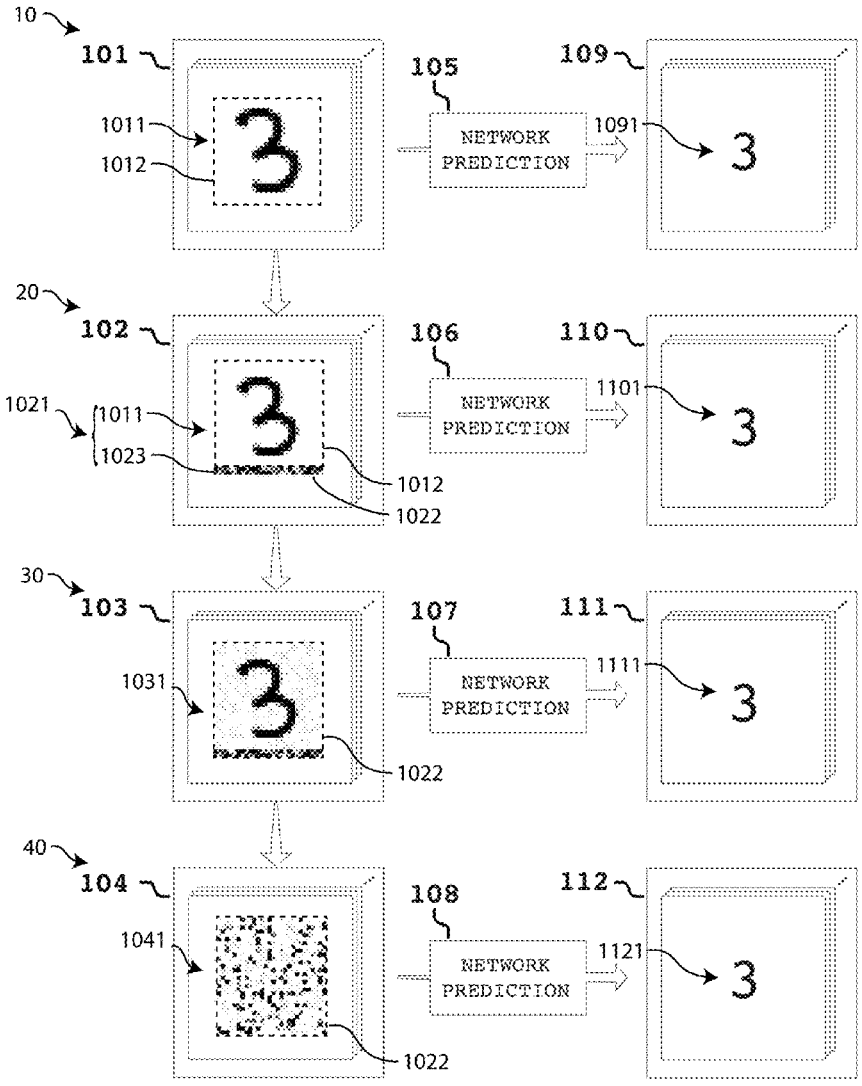
Publication Classification

(51) **Int. Cl.**
G06N 3/08 (2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/08*** (2013.01)

(57) **ABSTRACT**

Systems and methods are disclosed for encrypting data such that artificial neural networks may be trained directly on the encrypted data and may be performed directly on the encrypted data. For use with artificial neural networks, original feature sets of input vectors are programmatically subjected to a sequence of encryption steps: a method for fixed padding of the feature set with a randomly chosen set of values that are fixed across the feature set (adding a fixed number of dimensions to each input vector); a method for fixed perturbation of the data by perturbing of each element in the feature set (the value stored in each dimension of an input vector) such that all examples of the feature set are perturbed in the same way; a method that applies a fixed shuffle of the data elements in the feature set; and a method for applying convolutions (filters) and pooling (downsampling) to the shuffled data such that informational structure is preserved.



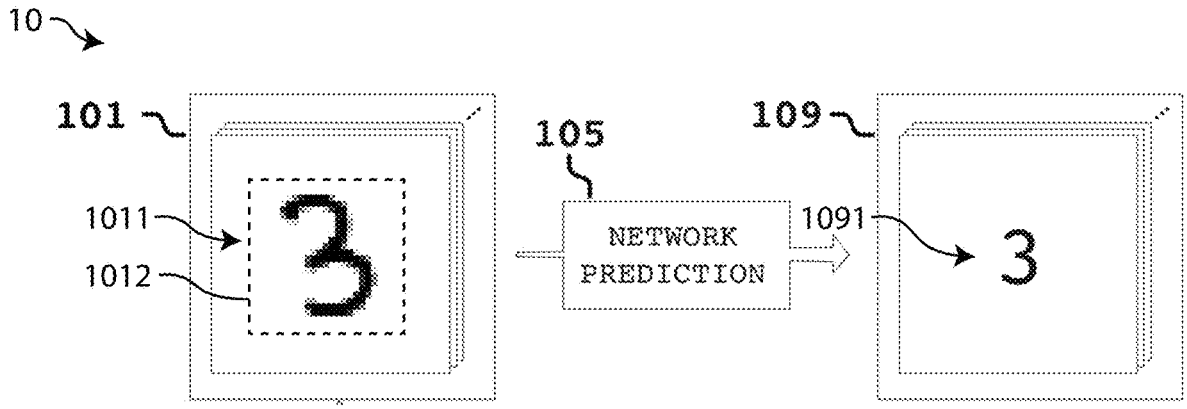


FIG. 1A (Prior Art)

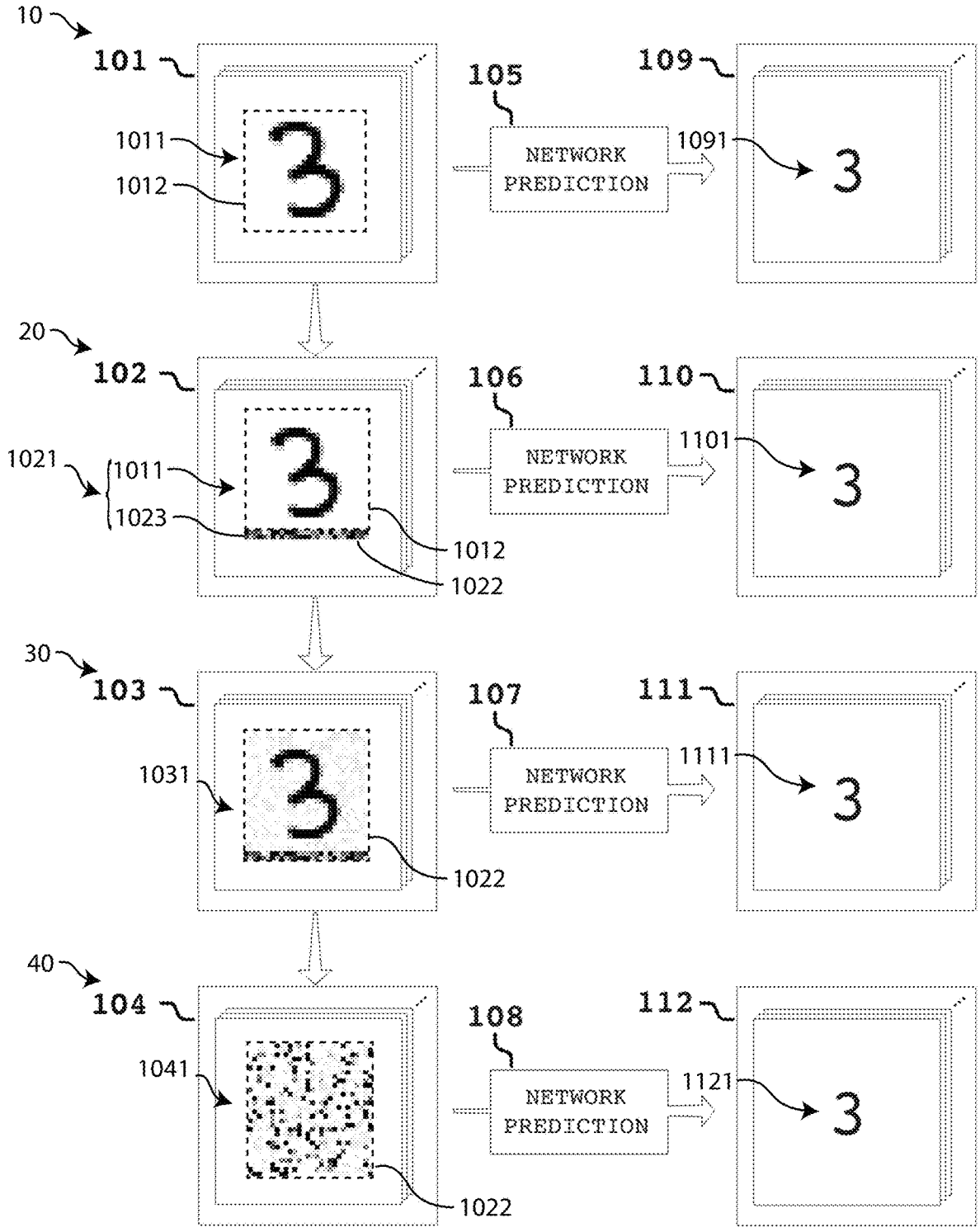

10
101
1011
1012
105
NETWORK PREDICTION
109
1091
20
102
1021
1011
1023
1012
1022
106
NETWORK PREDICTION
110
1101
30
103
1031
1022
107
NETWORK PREDICTION
111
1111
40
104
1041
1022
108
NETWORK PREDICTION
112
1121

FIG. 1B

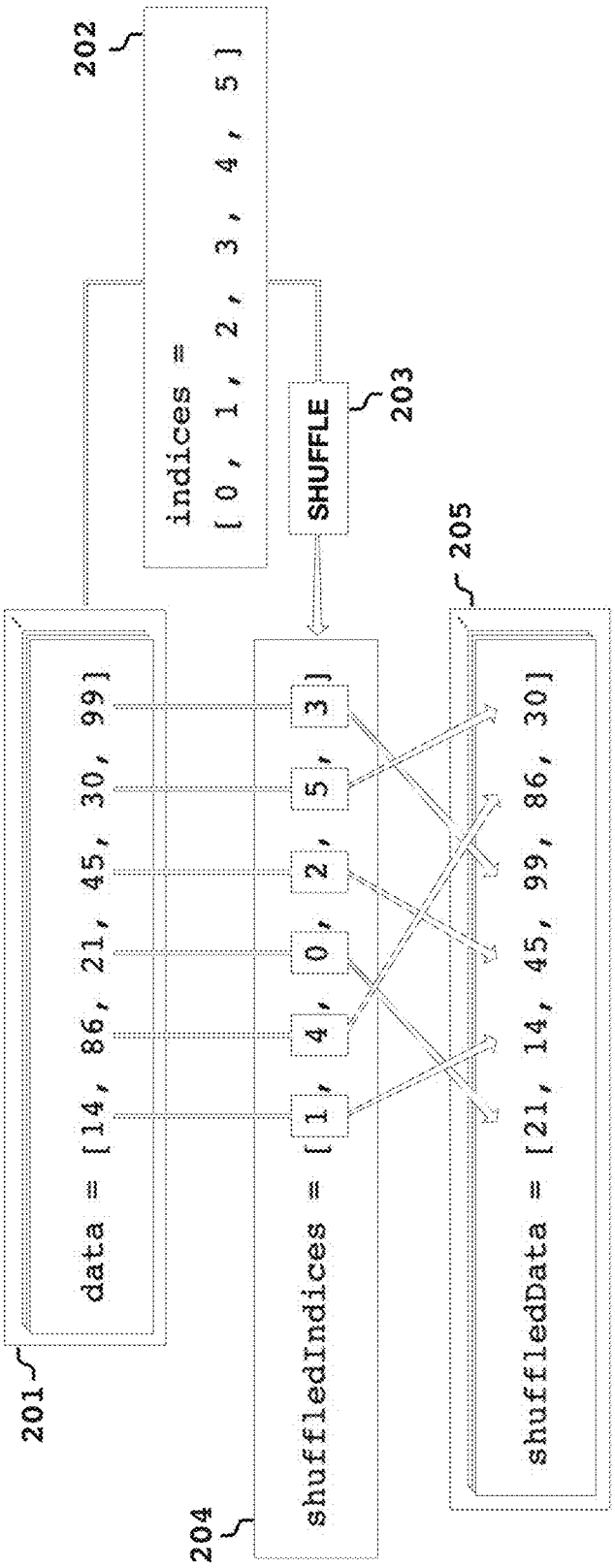

202
indices =
[0 , 1 , 2 , 3 , 4 , 5]
SHUFFLE
203
205
data = [14, 86, 21, 45, 30, 99]
shuffledIndices = [1 , 4 , 0 , 2 , 5 , 3]
shuffledData = [21, 14, 45, 99, 86, 30]
201
204

FIG. 2

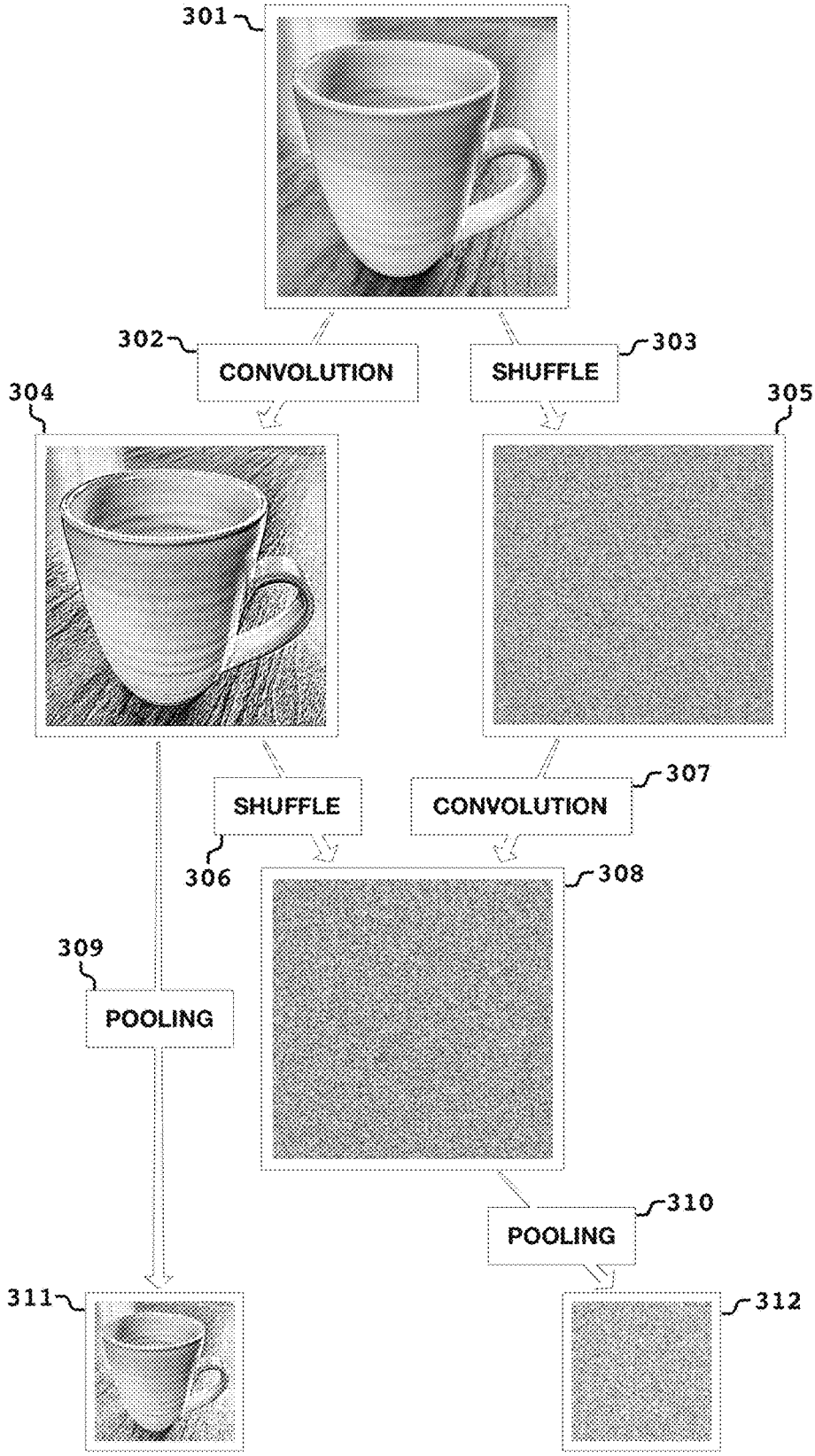

301
302
CONVOLUTION
303
SHUFFLE
304
305
SHUFFLE
306
CONVOLUTION
307
308
309
POOLING
310
POOLING
311
312

FIG. 3

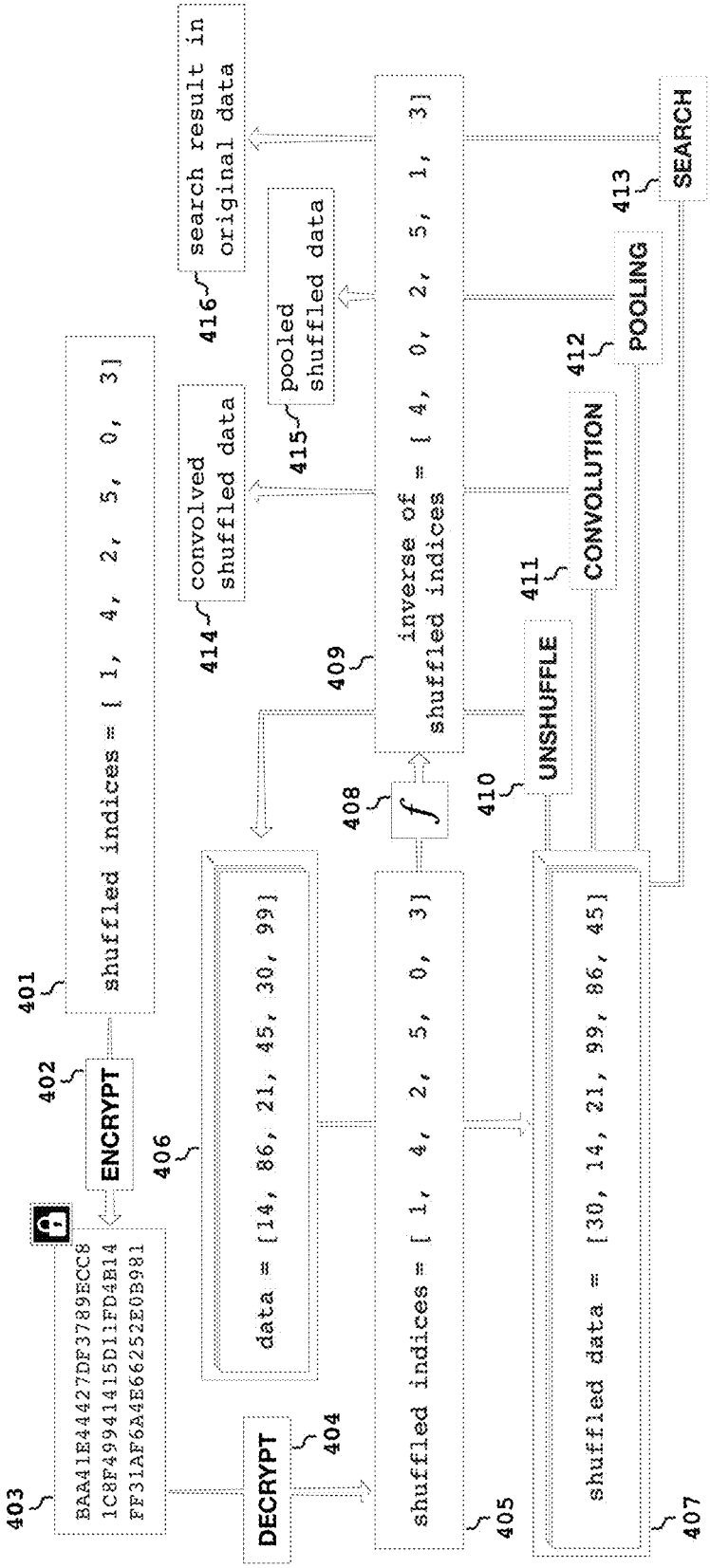

401
shuffled indices = [1, 4, 2, 5, 0, 3]
402
ENCRYPT
403
BAA1E44427DF3789ECC8
1C8F49941415D11FD4B14
FF31AF6A4E662S2E0B981
DECRYPT
404
406
data = [14, 86, 21, 45, 30, 99]
shuffled indices = [1, 4, 2, 5, 0, 3]
405
shuffled data = [30, 14, 21, 99, 86, 45]
407
408
f
409
inverse of shuffled indices = [4, 0, 2, 5, 1, 3]
410
UNSHUFFLE
411
CONVOLUTION
412
POOLING
413
SEARCH
414
convolved shuffled data
415
pooled shuffled data
416
search result in original data

FIG. 4

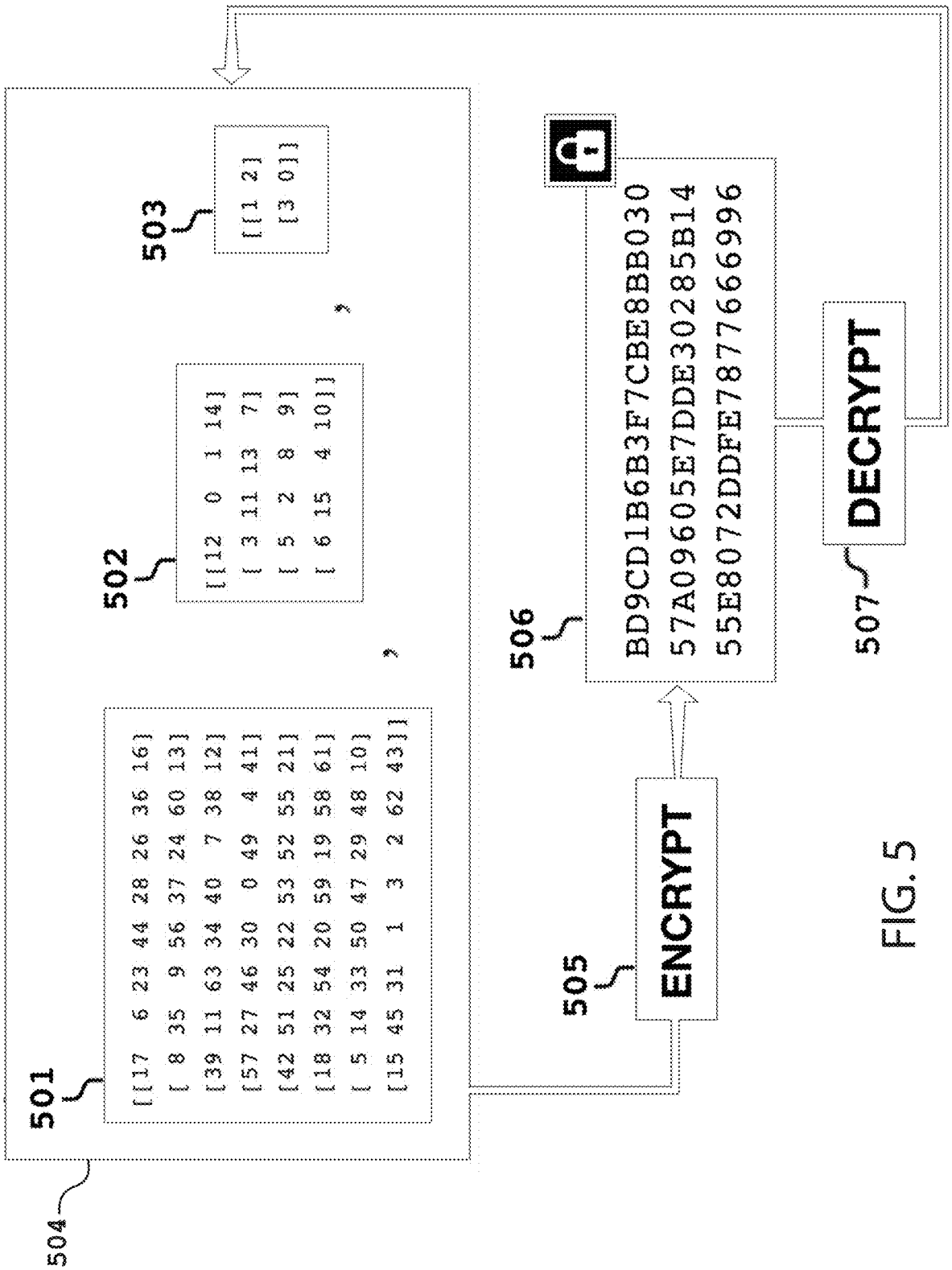

504
501
[[17 6 23 44 28 26 36 16]
[8 35 9 56 37 24 60 13]
[39 11 63 34 40 7 38 12]
[57 27 46 30 0 49 4 41]
[42 51 25 22 53 52 55 21]
[18 32 54 20 59 19 58 61]
[5 14 33 50 47 29 48 10]
[15 45 31 1 3 2 62 43]]
502
[[12 0 1 14]
[3 11 13 7]
[5 2 8 9]
[6 15 4 10]]
503
[[1 2]
[3 0]]
505
ENCRYPT
506
BD9CD1B6B3F7CBE8BB030
57A09605E7DDE302285B14
55E8072DDFE787766996
507
DECRYPT

FIG. 5

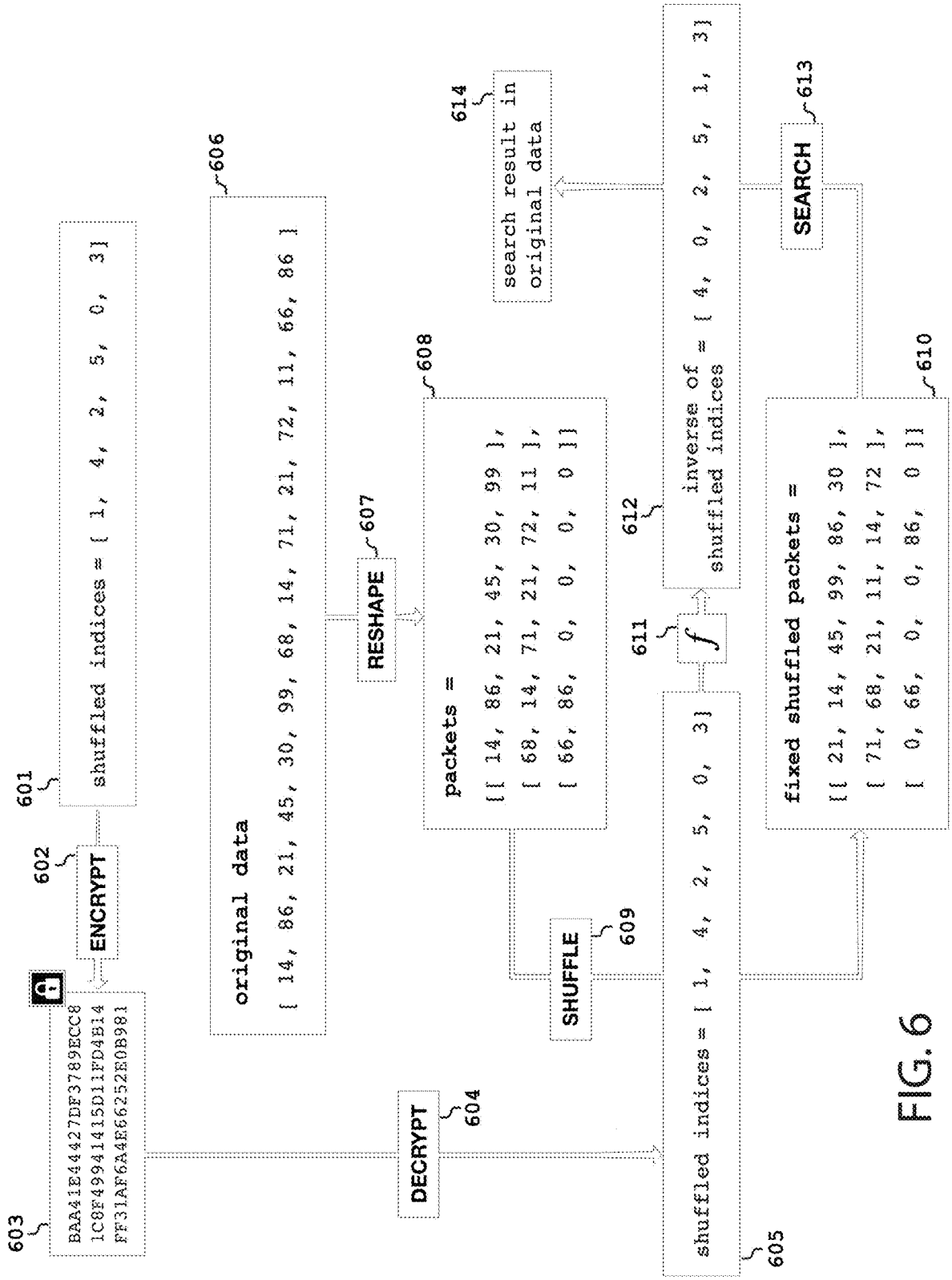

603
BAA41E44427DF3789ECC8
1C8F49941415D11FD4B14
FF31AF6A4E662526E0B981
601
shuffled indices = [1, 4, 2, 5, 0, 3]
602
ENCRYPT
606
original data
[14, 86, 21, 45, 30, 99, 68, 14, 71, 21, 72, 11, 66, 86]
607
RESHAPE
608
packets =
[[14, 86, 21, 45, 30, 99],
[68, 14, 71, 21, 72, 11],
[66, 86, 0, 0, 0, 0]]
609
SHUFFLE
604
DECRYPT
605
shuffled indices = [1, 4, 2, 5, 0, 3]
611
f
612
inverse of
shuffled indices = [4, 0, 2, 5, 1, 3]
614
search result in
original data
613
SEARCH
610
fixed shuffled packets =
[[21, 14, 45, 99, 86, 30],
[71, 68, 21, 11, 14, 72],
[0, 66, 0, 0, 86, 0]]

FIG. 6

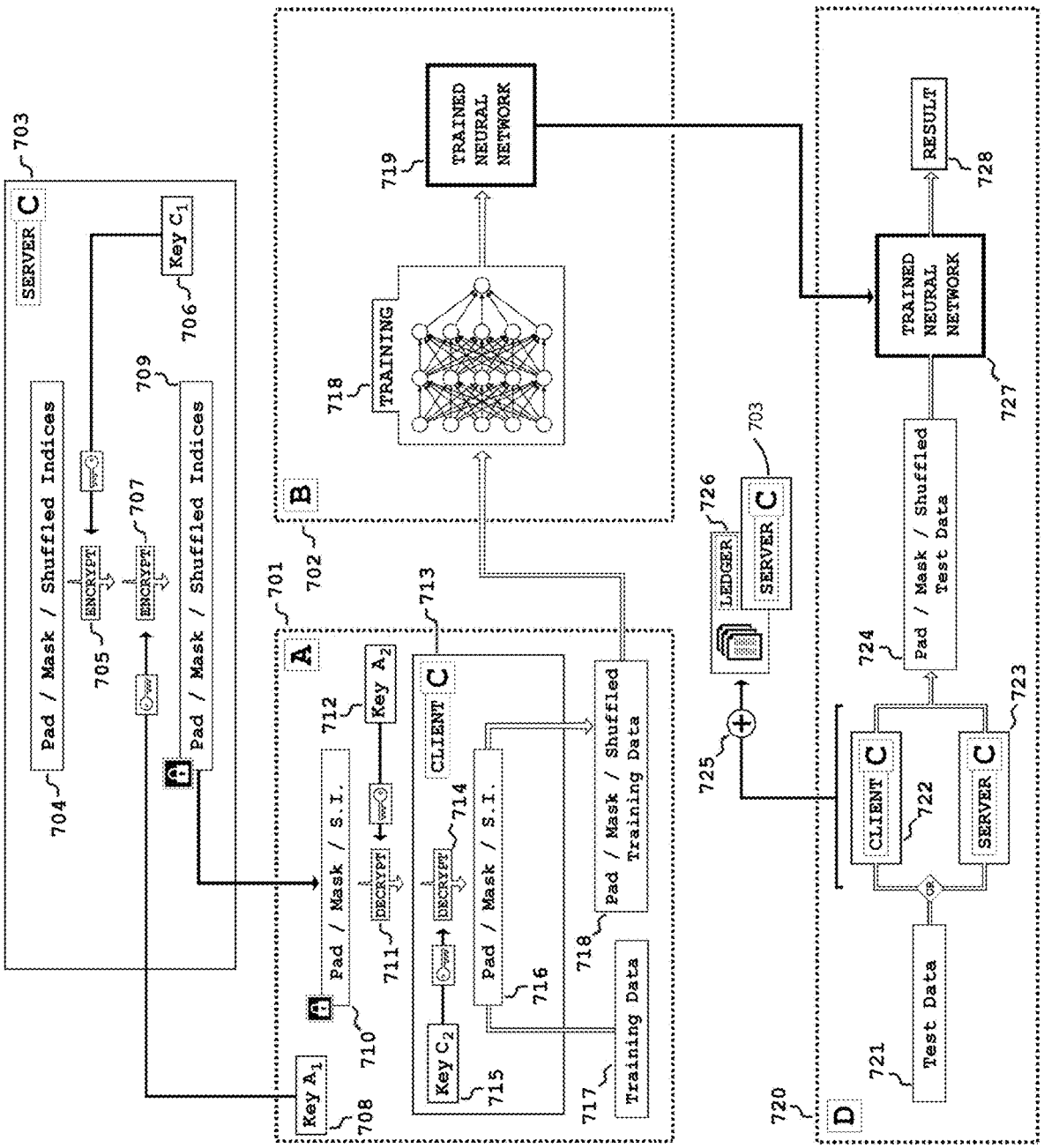

SERVER C
703
Pad / Mask / Shuffled Indices
704
705
ENCRYPT
707
Key C1
706
709
Pad / Mask / Shuffled Indices
A
701
B
702
TRAINED NEURAL NETWORK
719
TRAINING
718
Key A2
712
CLIENT C
713
Pad / Mask / S.I.
711
DECRYPT
714
Pad / Mask / S.I.
716
Pad / Mask / Shuffled Training Data
718
Training Data
717
Key A1
708
Pad / Mask / S.I.
710
Key C2
715
LEDGER
726
SERVER C
703
725
CLIENT C
722
SERVER C
723
Pad / Mask / Shuffled Test Data
724
TRAINED NEURAL NETWORK
727
RESULT
728
Test Data
721
D
720

FIG. 7

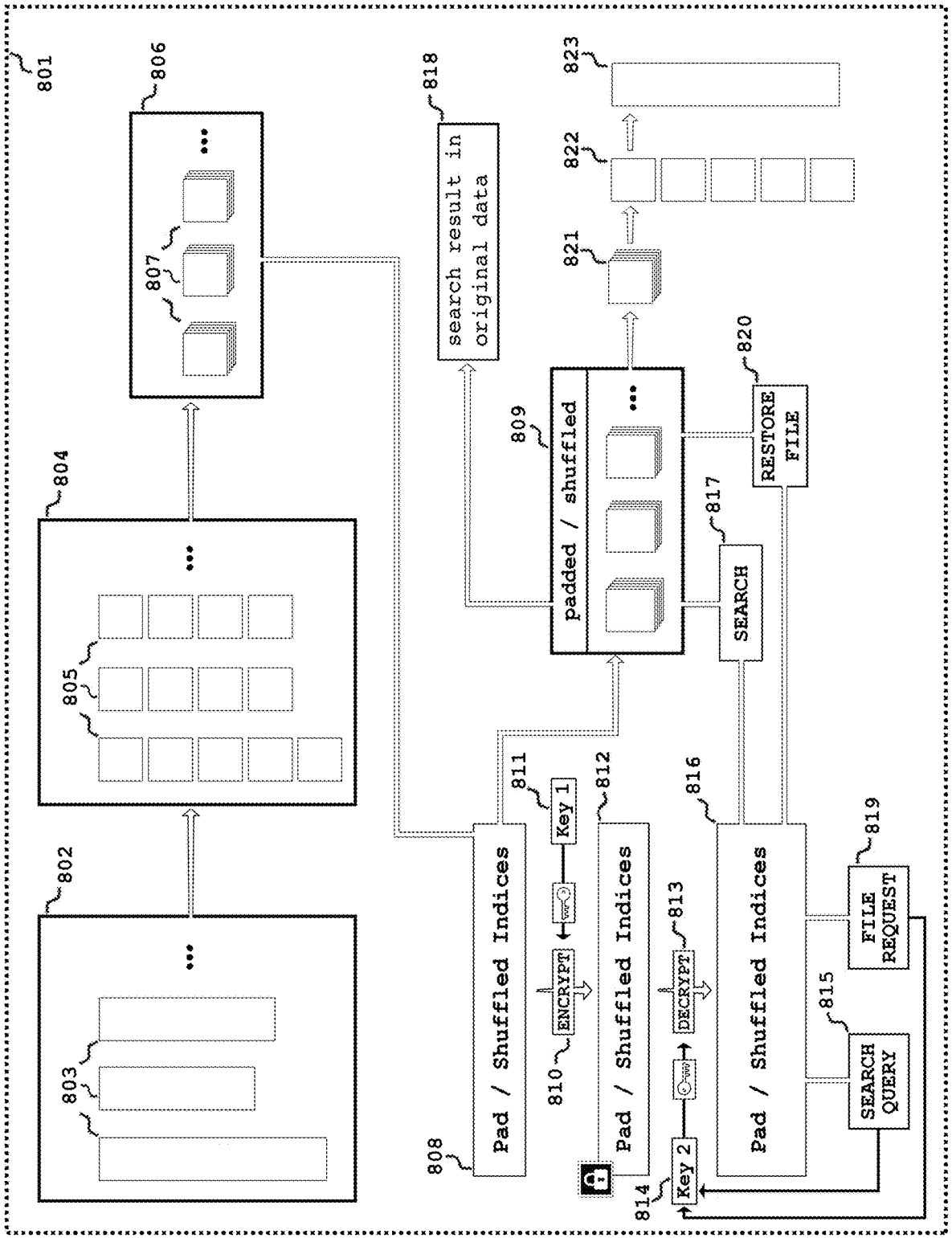

801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
Pad / Shuffled Indices
Key 1
ENCRYPT
Pad / Shuffled Indices
DECRYPT
Key 2
Pad / Shuffled Indices
SEARCH QUERY
FILE REQUEST
SEARCH
RESTORE FILE
padded / shuffled
search result in original data

FIG. 8

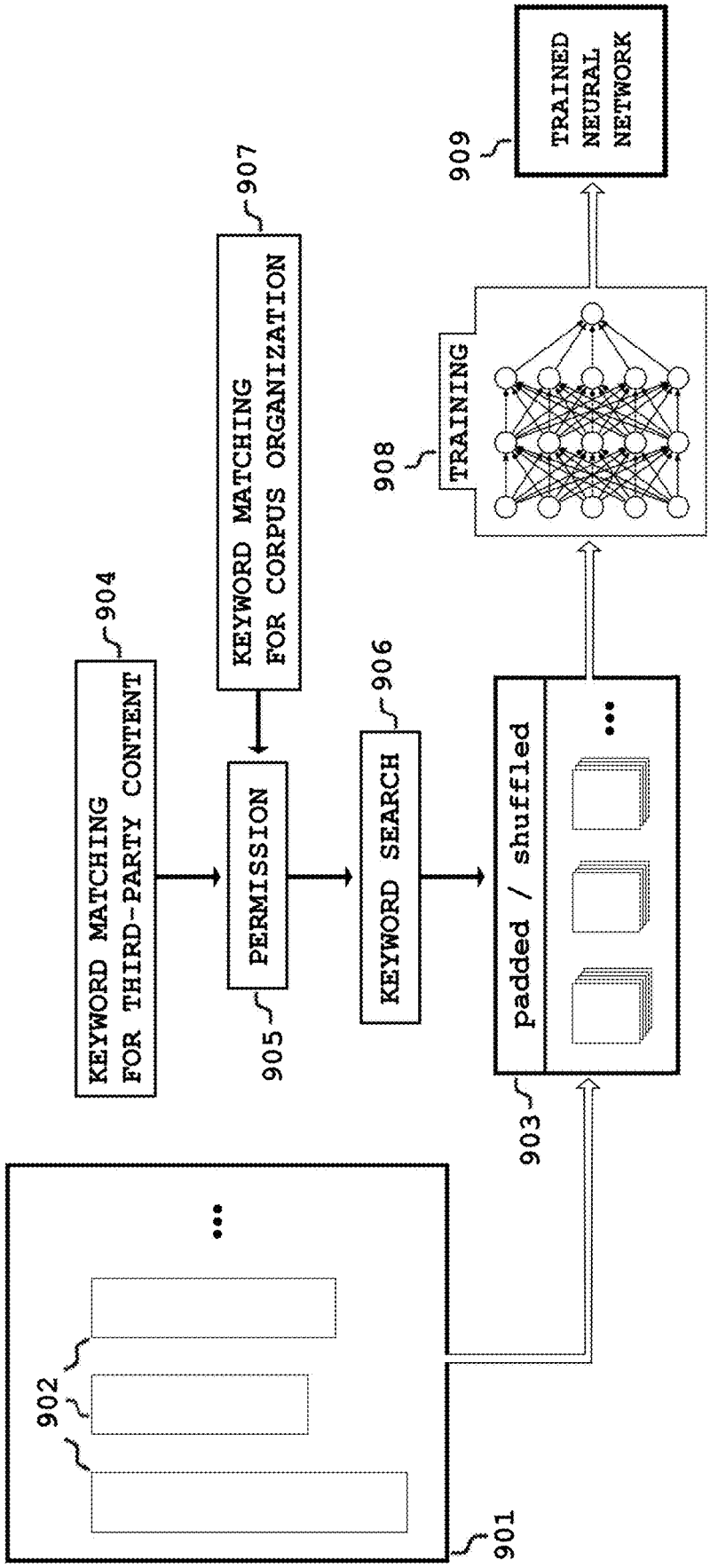

901
902
903
padded / shuffled
904
KEYWORD MATCHING FOR THIRD-PARTY CONTENT
905
PERMISSION
906
KEYWORD SEARCH
907
KEYWORD MATCHING FOR CORPUS ORGANIZATION
908
TRAINING
909
TRAINED NEURAL NETWORK

FIG. 9

METHOD AND SYSTEM FOR SECURELY STORING DATA FOR USE WITH ARTIFICIAL NEURAL NETWORKS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 63/155,210, filed Mar. 1, 2021, the contents of which is hereby incorporated by reference in its entirety.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The following relates to data security in the fields of artificial neural networks and algorithmic search.

Discussion of the Background

[0003] Artificial neural networks have a set of inputs, a number of 'hidden' layers of units, and a set of outputs, where the layers are interconnected with adjustable-weight connections. Training an artificial neural network is an iterative process that determines the weights of each connection by presented the neural network with a large training set of input vectors with known outputs in order to mathematically optimize the set of adjustable weights between units. After the network learns a set of weight adjustments based on the training dataset, a separate validation dataset is then tested to determine the accuracy of the trained network in determining outputs from presented input vectors. When the network is trained to an appropriate level of accuracy, the trained network may then be used to determine outputs in automated fashion from test input vectors for which outputs are not yet known.

[0004] FIG. **1**A illustrates a prior art method (**10**) for using an artificial neural network (ANN) using a dataset (**101**). As is known in the field, ANN (**105**) may be trained to determine the node weights of the ANN by comparing known inputs from dataset (**101**) with predictions (**109**) of the ANN. In general, dataset (**101**) consists of a plurality of elements, which may be, for example and without limitation, be a digital image. Once ANN (**101**) is trained, the ANN may be used, as illustrated in FIG. **1**A, to make predictions of elements which may or may not be part of dataset (**101**).

[0005] Thus, for example and without limitation, FIG. **1**A shows an exemplary image (**1011**) of dataset (**101**) as being a digital image of a handwritten number 3. Using method (**10**), as known in the prior art, dataset (**101**) is provided to trained network (**105**), which generates a set of corresponding predictions (**109**), with one prediction for each image of dataset (**101**). FIG. **1** shows, for example, that neural network (**105**) accepts image (**1011**), which is an image of a handwritten number 3, and generates a prediction (**1091**), which is the numeral "3".

[0006] Addition of convolutional and pooling layers to the neural network can improve accuracy and efficiency in tasks such as image interpretation (determining outputs based on input vectors that represent images or video data). Convolutional layers apply a set of image transformations (such as sharpening or edge enhancement) based on the relationships between neighboring pixels, and pooling layers reduce the size of the convolved image vectors, downsampling neighboring pixels by taking the maximum, average, or other derived value from a neighboring group of pixels.

[0007] While there is an evolving set of neural network techniques and substantial cloud computing resources available to train networks on new data, data sources that would be suitable for such training are often proprietary or sensitive in nature. Organizations that possess such proprietary or sensitive datasets may not have the necessary technical expertise or local computing resources to train neural networks. Training modern artificial neural networks with large datasets typically relies on cloud computing resources, because such tasks would be computationally intractable using on-premises computing resources. Organizations with proprietary or sensitive datasets may be reluctant to share their data to cloud repositories for the purpose of training artificial neural networks. Transmission of sensitive data to cloud computing resources in an unencrypted form presents a security risk that is impossible to mitigate. Additionally, once an artificial neural network is trained using proprietary data, that network can be used on test data from any source. This leads to a devaluing of proprietary training data that may have been difficult or expensive for an organization to accumulate. These tensions have slowed development of artificial neural networks trained on proprietary or sensitive data.

[0008] One proposed solution to some of these problems of data security in training artificial neural networks is known as 'federated learning.' Federated learning involves a central cloud server that is connected to a number of software nodes that can run behind the security firewalls of various organizations or on individual devices. An aggregate model is sent from the server to the nodes, and the nodes perform training on data locally. The resultant local adjustments to model weights are then sent back to the central server, where these weights are averaged with the results of other local training sessions. While federating learning has advantages in areas such as mobile privacy, where lightweight training can occur on a device using a local mobile user's data, with only weight updates sent back to the central server, it fails to solve several of the core issues of concern for organizations with proprietary or sensitive data. In the framework of federated learning, node software from an external source must be deployed behind the organization's firewall, which is a security risk and involves unverifiable trust steps (for example, the organization must trust that the externally-sourced node software will not send any training data to the central server during communication). The distributed nature of federated learning, while a strength in some applications, is also a weakness in the specific application of machine learning to proprietary data sources. The devaluing of an organization's proprietary data remains an intrinsic flaw of artificial neural networks trained using federated learning.

[0009] Methods for programmatic search, namely identification of the presence and location of specific elements within a corpus of information stored in any manner of computer memory, have been extensively developed and refined. One notable limitation of programmatic search is encountered when the corpus of information to be searched is encrypted by conventional methods. In conventional methods for encryption, raw data is converted by means of one or more cryptographic keys and an algorithm that creates an encrypted version of raw data that is typically not the same size as the raw data in memory. If a user or system

intends to perform programmatic search on a corpus of encrypted information, the entire corpus must be decrypted to the raw (unencrypted) form of the corpus, prior to algorithmic search. For a large corpus of information in computer memory or distributed solutions, such as web- or cloud-based computing, the process of decryption for search is time- and resource-intensive. For a corpus of information on a computer system, cloud system, electronic communication system, or social media system that is encrypted for privacy purposes, it is not possible to perform functions such as matching advertisements to content because the content is encrypted by conventional methods that prevent the necessary search functions required to make such a match. One solution in common use that allows for algorithmic search of data stored in cloud systems is to store data at rest in unencrypted form, with password-protection applied to user access to the cloud system. This approach suffers from the known security vulnerabilities inherent to any system with data stored in unencrypted form

[0010] Thus, there is a need in the art for a method and apparatus that permits for the training and for the use of neural networks that allows for owners of the data used for training the neural network to retain control of their data. Such a method and apparatus should be easy to implement, secure, and be computationally efficient.

BRIEF SUMMARY OF THE INVENTION

[0011] In one aspect of the invention, a method for using modified data with a neural network is provided, where the method includes determining an algorithm to modify a plurality of examples, where each example of the plurality of examples includes an array of values, and where each example is a training example or a test example, training the neural network, and then forming predictions using the trained neural network. In certain embodiments, the training the neural network includes obtaining a plurality of training examples, modifying each training example of said plurality of training examples according to said algorithm to form a plurality of modified training examples, and forming a trained neural network by training a neural network with said plurality of modified training examples. In certain embodiments, the forming predictions using the trained neural network includes accepting a test example, modifying the test example according to said algorithm to form a modified test example, and forming a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input. The method thus provides that the same algorithm modifies each training example and the test example. In various embodiments: 1) the algorithm is encrypted prior to forming predictions using the trained neural network to form an encrypted algorithm, and the forming predictions using the trained neural network further includes accepting said encrypted algorithm and decrypting said encrypted algorithm; 2) the algorithm includes one or more of a) a first pad of values which are appended to the examples, b) a plurality of perturbation functions applied to each example; and c) an index shuffling applied to each example; 3) the training the neural network is performed by two or more parties; and/or 4) a ledger tracks the use of the neural network.

[0012] In another aspect of the invention, a system for using modified data with a neural network is provided, where the system includes networked memory and processors programmed to: determine an algorithm to modify a plurality of examples, where each example of the plurality of examples includes an array of values, and where each example is a training example or a test example; train the neural network, and form predictions using the trained neural network. The training of the neural network includes the processors programmed to: obtain a plurality of training examples, modify each training example of said plurality of training examples according to said algorithm to form a plurality of modified training examples, and form a trained neural network by training a neural network with said plurality of modified training examples. The forming of predictions includes the processors programmed to: accept a test example, modify the test example according to said algorithm to form a modified test example, and form a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input. The system thus uses the same algorithm is used to modify each training example and the test example. In various embodiments the memory and processor are programmed such that: 1) the algorithm is encrypted prior to forming predictions using the trained neural network to form an encrypted algorithm, and the forming predictions using the trained neural network further includes accepting said encrypted algorithm and decrypting said encrypted algorithm; 2) the algorithm includes one or more of a) a first pad of values which are appended to the examples, b) a plurality of perturbation functions applied to each example; and c) an index shuffling applied to each example; 3) the training the neural network is performed by two or more parties; and/or 4) a ledger tracks the use of the neural network.

[0013] In one aspect of the invention, a method for using modified data with a neural network is provided, where the neural network is trained using a plurality of modified training examples, where each training example of the plurality of training examples is modified using an algorithm. The method includes accepting a test example, accepting the algorithm used for modifying each training example of the plurality of training examples, modifying the test example using the algorithm, and forming a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input. The method is such that the same algorithm is used to modify each training example and the test example. In various embodiments: 1) the algorithm is encrypted, and the forming predictions using the trained neural network further includes accepting said encrypted algorithm and decrypting said encrypted algorithm; 2) the algorithm includes one or more of a) a first pad of values which are appended to the examples, b) a plurality of perturbation functions applied to each example; and c) an index shuffling applied to each example; and/or 3) a ledger tracks the use of the neural network.

[0014] In another aspect of the invention, system for using modified data with a neural network, where the neural network is trained using a plurality of modified training examples, where each training example of the plurality of training examples is modified using an algorithm provided, where the system includes networked memory and processors programmed to: accept a test example, accept the algorithm used to modify each training example of the plurality of training examples, modify the test example using the algorithm, and form a prediction from the output of the trained neural network by providing said trained

neural network with the modified test example as input. The system is such that the same algorithm is used to modify each training example and the test example. In various embodiments the memory and processors are programmed such that: 1) the algorithm is encrypted, and the forming predictions using the trained neural network further includes accepting said encrypted algorithm and decrypting said encrypted algorithm; 2) the algorithm includes one or more of a) a first pad of values which are appended to the examples, b) a plurality of perturbation functions applied to each example; and c) an index shuffling applied to each example; and/or 3) a ledger tracks the use of the neural network.

[0015] These features together with the various ancillary provisions and features which will become apparent to those skilled in the art from the following detailed description, are attained by the method and system of the present invention, preferred embodiments thereof being shown with reference to the accompanying drawings, by way of example only, wherein:

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

[0016] FIG. **1**A illustrates a prior art method for predicting using artificial neural networks;

[0017] FIG. **1**B illustrates a number of inventive methods for predicting using artificial neural networks;

[0018] FIG. **2** illustrates the method of fixed shuffle processing of an input dataset;

[0019] FIG. **3** illustrates methods to apply convolutions (filters) and pooling (downsampling) to datasets that have been transformed by fixed padding, fixed perturbation, and fixed shuffle processes;

[0020] FIG. **4** illustrates methods to securely utilize shuffled indices and the inverse of shuffled indices to perform functions such as unshuffling, convolution, pooling, or search to fixed shuffled datasets;

[0021] FIG. **5** illustrates methods to securely store and transmit packages of shuffled indices corresponding to the different sizes of data arrays created by pooling;

[0022] FIG. **6** illustrates methods to prepare data for storage and transmission in fixed shuffled packets for use with programmatic search;

[0023] FIG. **7** illustrates a system in which multiple parties may use secure data for training and implementation of artificial neural networks;

[0024] FIG. **8** illustrates a system wherein files on a computer system or cloud service may be stored securely in a format that allows for search and file restoration by credentialed users; and

[0025] FIG. **9** illustrates a system wherein files, messages, posts, notes, or other electronic communications on a mobile, online, Web-based, or cloud-based electronic communication system or computer system or cloud service may be stored securely in a format that allows for keyword matching search and secure neural network training by permissioned entities.

[0026] Reference symbols are used in the figures to indicate certain components, aspects or features shown therein, with reference symbols common to more than one figure indicating like components, aspects or features shown therein.

DETAILED DESCRIPTION OF THE INVENTION

[0027] FIG. **1**B illustrates inventive methods (**20**), (**30**), and (**40**) for predicting using artificial neural networks. The inventive methods may be understood by comparison to prior art method (**10**), as described above and which is also included in FIG. **1**B. As discussed above, prior art method (**10**) trains ANN (**105**) using dataset (**101**). Methods (**20**), (**30**), and (**40**) differ from prior art method (**10**) in that each element of dataset (**101**) is first modified, as described subsequently, to form a corresponding modified dataset (**102**), (**103**), and (**104**), which is used to train corresponding artificial neural network (**106**), (**107**), and (**108**), and which may then be used to form predictions (**110**), (**111**), and (**112**), respectively. The present invention will be described, without limitation, as having dataset elements that are images each having the same number of horizontal and vertical pixels, and with each pixel having the same number of bits of depth.

[0028] For illustrative purposes, dataset (**101**) is shown in FIG. **1** as the standard dataset known as "MNIST" (http://yann.lecun.com/exdb/mnist/), where each image in the dataset is a digitized image of a handwritten digit. By way of example, each image of dataset (**101**) has a dimension of 28 by 28 (x by y) pixels, where the outline is indicated by dashed line (**1012**) that is not part of the dataset, and each pixel has a grayscale value with 8-bit depth.

[0029] The process of training neural networks using input datasets and using trained neural networks to generate predictions is well-known in the prior art. The following discussion inventively uses these methods, as well as various methods of using neural networks on modified or encrypted datasets.

[0030] An example of a novel method (**20**) illustrates the use of a dataset (**102**) for training and prediction from an artificial neural network (**106**) to produce a set of predictions (**110**). Individual images of dataset (**102**) are modified images of dataset (**101**), where the modification adds a "fixed padding" to the images of dataset (**101**). The fixed padding, in this illustrative example, is a randomly generated block of pixels having a rectangular shape with at least one dimension corresponding to a dimension of images in dataset (**101**).

[0031] An exemplary image (**1021**) of dataset (**102**) includes image (**1021**) having an outline (**1022**). Image (**1021**) includes image (**1011**) having outline (**1012**), and the fixed padding (**1023**) of size 28 by 2 (x by y) with 8-bit depth that is positioned below image (**1011**). Each image of dataset (**102**) thus includes a different image from dataset (**101**) that is combined with an identical 28×2 pixel fixed padding provided below the image of dataset (**101**). Specifically, the padding of each image of dataset (**102**) is the same and is thus referred to as having a "fixed padding" or as having the "application of a fixed padding."

[0032] Using method (**20**), artificial neural network (**106**) is trained using the fixed padded dataset (**102**), providing the resulting set of predictions (**110**). The accuracy of the predictions obtained using method (**20**) by adding fixed padding to dataset (**101**) does not differ substantively from that obtained from method (**10**) because the padding set is invariant across the elements of the dataset, and therefore these invariant data have only minor contributions to the adjustments of network weights during training and these invariant data therefore do not contribute substantively to

the prediction (**110**) of the trained network. Thus, for example, neural network (**106**) accepts image (**1021**), which is a modified image of a handwritten number 3 composed of the original image of the handwritten number 3 (**1011**) with the randomly generated padding data (**1023**) appended, and generates a prediction (**1101**), which is the numeral "3," where prediction (**1101**) is the same as prediction (**1091**). Within the scope of the present invention, an arbitrary size and shape of fixed padding data may be used, and the position of the fixed padding data may be in any chosen relation to the shape of the dataset to be padded, such that any number of blocks of fixed padding data may be appended to the parent dataset, external to, or within, the dimensions of the parent dataset. The choice of fixed padding data is not limited to sizes and shapes of fixed padding data that retain the original tensor rank of the parent dataset, and fixed padding data may be chosen such that appending the fixed padding data raises the rank of the modified tensor above that of the parent dataset—for example, a parent dataset tensor of rank 3 could be converted to a tensor of rank 4 or higher by appending fixed padding data of the appropriate shape. In additional embodiments, the shape of the original tensor may be cropped prior to appending the fixed padding data such that the shape of the resulting fixed padded tensor is not different from the original tensor shape before the application of cropping and appending of the fixed padding. In additional embodiments, the size of the original tensor before padding is larger than is needed for machine learning training (e.g., original examples have a border of invariant values), so application of cropping followed by fixed padding retains the original tensor shape but does not remove any useful training information from the original dataset. The choice of fixed padding may be determined by generation of a random block of data of arbitrary size and shape that matches the bit depth of the parent dataset, as illustrated in (**102**), or the statistical distribution of randomly generated values may be constrained by a priori considerations such as the statistical distribution of values in the parent dataset.

[0033] An example of an additional novel method (**30**) illustrates the use of a dataset (**103**) for training and prediction from an ANN (**107**) to produce a set of predictions (**111**). Individual images of dataset (**103**) are modified images of dataset (**102**), with each pixel of each image of dataset (**102**) being subjected to a transformation that perturbs the pixel value. In one embodiment, the transformation is performed by applying a mathematically defined "perturbation function" to each pixel of every image. There are thus as many perturbation functions as there are pixels in an image, and the perturbation functions may be stored in a "perturbation array" which is the same size as each image. The transformation may occur, for each image, by applying the perturbation function in each of the perturbation array locations to the value of the corresponding pixel, and the same perturbation array is applied to each of the images. In general, the perturbation functions may be different or they may be the same. In the illustrative example (**30**), each pixel is perturbed by a small, randomly determined, amount. By way of example, an exemplary image (**1031**) of dataset (**103**) has the size of image (**1021**).

[0034] In an illustrative example, each image of dataset (**103**) takes an image of dataset (**102**) and adjusts each pixel by a randomly chosen amount, increasing or decreasing the value of each parent pixel from dataset (**102**) by a random amount varying, in the presented example, between a maximum range of a decrease of 5% to an increase of 5%, thus yielding a fine amount of light gray pixel random perturbations as shown in image (**1031**)—in practice, a smaller range of perturbations may be applied, and the range of (−5% to +5%) is used in this example so that the perturbations are visible in the illustration. In the illustrative example, the array of perturbation functions are applied by storing the randomly-chosen adjustment percentage values in a data array corresponding to the shape of the images of the parent dataset (**102**), in this example 30 by 28 pixels, and this stored perturbation array is applied by increasing or decreasing the value of the corresponding pixels in the example image. Application of the array of perturbation functions to each pixel of the image is then applied in identical fashion to each of the images of the parent dataset (**102**), yielding the resulting dataset (**103**). Because this perturbation of the dataset is fixed for each element of the dataset, this process will be referred to henceforth and without limitation, as "fixed perturbation" or "application of a fixed perturbation". The choice of perturbation functions stored in the fixed perturbation array and applied uniformly to each of the elements in the parent dataset (**102**) may be a randomly chosen adjustment value between a fixed negative percentage of the original value and a fixed positive percentage of the original value, as illustrated, or it may be, without restriction, any array of linear or nonlinear mathematical perturbation functions acting on the original values. Values transformed by the fixed perturbation method that would exceed the bounds of the range of values in use (for example, the bit depth of values in use) are, in preferred embodiments, transformed by means established in the digital signal processing art such as constraint to maximum or minimum values (clipping), reflection to values within the established range, wrapping around from one end to the range to the other, or other established digital signal processing methods including compression or limiting. The choice of positive or negative range limitations to any randomly chosen perturbation values or the choice of mathematical transformation may be adjusted according to considerations such as the statistical distribution of values in the parent dataset. The fixed perturbation process, in preferred embodiments, may be constrained to a choice of perturbation functions that minimally alter the values stored in the original elements of dataset (**102**), because very large fixed perturbations may negatively impact ANN accuracy.

[0035] Method (**30**) yields predictions (**111**) that do not differ substantively in accuracy from those of methods (**10**) or (**20**) despite the large (up to −/+5% perturbation) used for purposes of graphical illustration in the presented example. Thus, for example, neural network (**107**) accepts image (**1031**), which is a modified image of a handwritten number 3, and generates a prediction (**1111**), which is the numeral "3," where prediction (**1111**) is the same as prediction (**1091**) and (**1101**). It will be apparent to those skilled in the art that the chosen range of fixed perturbations would have an impact on model accuracy when larger perturbations are introduced, and thus an appropriate extent of fixed perturbations would be chosen for a given training task.

[0036] In other embodiments, the bit depth or data type of the source data is transformed to facilitate the fixed perturbation process. For example, a dataset with 8-bit depth may be transformed to the corresponding dataset with 16-bit depth, or any other choice of bit depth higher or lower than

the starting value, by standard means known in the art, or by conversion of one data type to another data type (e.g. conversion of integer data to floating-point data). Such bit-depth or type transformation allows for finer gradation adjustments of the data by the fixed perturbation process described herein, and may also be used to further prevent determination of the fixed shuffle described below by decoupling the bit depth or data type of the unshuffled source dataset and that of the fixed shuffled dataset.

[0037] An example of an additional novel method (**40**) illustrates the use of a dataset (**104**) for training and prediction from an ANN (**108**) to produce a set of predictions (**112**). Individual images of dataset (**104**) are images of dataset (**103**) that are modified by a mathematical transformation in which each data point (pixel) in each image in dataset (**103**) is subjected to a process of "fixed shuffling." Fixed shuffling rearranges the data in each image of dataset (**103**) by reordering the position of the pixels and storing the shuffled pixels in a corresponding image of dataset (**104**). The shuffling of the pixels is randomly determined, but each image of dataset (**103**) is shuffled according to the same movements of pixel position to form the corresponding image of dataset (**104**). Information on how to shuffle the pixels of each image of example in a dataset containing a plurality of images or examples is stored as a set of "shuffled indices." This shuffling process is thus referred to as a "fixed shuffling" or an "application of a fixed shuffle" to the dataset, yielding a fixed shuffled dataset (**104**). Further details of the fixed shuffling method will be provided by reference to FIG. **2**.

[0038] Method (**40**) yields predictions (**112**) that do not differ substantively in accuracy from those of methods (**10**), (**20**), or (**30**). Thus, for example, neural network (**108**) accepts image (**1041**), which is a modified image of a handwritten number 3, and generates a prediction (**1121**), which is the numeral "3", where prediction (**1121**) is the same as prediction (**1091**), (**1101**), and (**1111**).

[0039] It will be understood to those skilled in the art that the methods illustrated in FIG. **1**B may be applied to any dataset of tensors of any degree (any number of input dimensions with any shape, and consisting of any suitable data type) and is not restricted to the example of two dimensional (x by y pixel) images shown in FIG. **1**B for purposes of illustration. The process that is shown for illustrative purposes in FIG. **1**B as a sequential application of fixed padding, followed by fixed perturbation, and then followed by fixed shuffling is one embodiment of the present invention, but these three processes (fixed padding, fixed perturbation, and fixed shuffling) may be applied alone or in different combinations or orders (for example, application of fixed padding then fixed shuffling, without fixed perturbation; or application of fixed shuffling alone), and also may be applied iteratively (for example, application of fixed shuffling, then fixed padding, then fixed perturbation, then a second round of fixed shuffling).

[0040] FIG. **2** illustrates the method of fixed shuffle processing of an input dataset. FIG. **2** shows, for illustrative purposes, displaying a small dataset (**201**) consisting of a 1-D array of 6 values for each example in the dataset (a rank 2 tensor consisting of a 6-dimensional vector by any number (n) examples). Taking the first example of the dataset (**201**) shown at the top of the stack, the indices of these 6 values are shown in (**202**), beginning, per usual convention in the art, with index 0. These indices are stored in an array (**202**) of the same size of the array shown in (**201**), and this array of indices is then subjected to a random shuffling algorithm (**203**) once, resulting in a shuffled index array (**204**) in which the index values are now found in a new position. The shuffled index array (**204**) is stored in memory and used to apply a fixed shuffle to each element of the full dataset (**201**), resulting in a fixed shuffled dataset (**205**). The fixed shuffle is applied by moving each value in each element of the dataset (**201**) to the new index position in (**205**) based on the index value found in the fixed shuffle index array (**204**). By inspection of FIG. **2**, each value in the original array in (**201**) can be traced to its new position in (**205**) by way of the index value found in (**204**). For example, the value '86' in (**201**) is found at index=1 in (**201**), so the shuffled index value of '4' is obtained from the position of index=1 in (**204**), indicating that the value '86' should be placed in position index=4 in the shuffled dataset (**205**). As described above, the process of generating the shuffled index array (**204**) by shuffling (**203**) of the appropriately-shaped index array (**202**) is performed once for an entire dataset, and the identical shuffling process as illustrated in (**201**, **204**, **205**) is applied to each element of the dataset, yielding a fixed shuffle. The shuffled index array (**204**) therefore represents a lookup table that can be encrypted by conventional methods, along with the corresponding lookup tables for fixed padding and fixed perturbation as described above. It will be apparent to those skilled in the art that the process of fixed shuffling can be applied to any dataset of tensors of any rank, either by the direct shuffling of arrays of the appropriate number of dimensions (e.g., the 1-D array shown in FIG. **2** or the 2-D image array shown in FIG. **1**A, or arrays of any number of dimensions), or by first flattening a higher dimensional array to a 1-D array with the application of the 1-D array shuffling method as shown in FIG. **2**, followed by reshaping back to the original number of array dimensions.

[0041] FIG. **3** illustrates methods to apply convolutions (filters) and pooling (downsampling) to datasets that have been transformed by fixed padding, fixed perturbation, and fixed shuffle processes as described above. The feature manipulations known as convolutions and pooling are used in a subclass of ANNs, known as convolutional neural networks, to achieve improved performance in image-recognition and other tasks involving datasets in which individual values in data elements may have spatial relationships with other individual values. Convolution involves application of a convolution matrix filter, also known in the art as a kernel, to an original image, with the kernel representing a small grid of values that are used to transform the value of a pixel based on a mathematical formula using the values in the kernel grid and the values of pixels surrounding the pixel of interest. The result of the convolution process by standard methods known in the prior art is shown for purposes of illustration in FIG. **3**, with transformation of an image (**301**) into an edge-enhanced image (**304**) by application of a standard convolution using a kernel that has the visual effect of edge-enhancement. Pooling involves creation of a smaller image from a larger one by using neighboring pixel values (e.g., a 2×2 grid of values) in the parent image to generate a single value in the resulting smaller image by taking, for example, the average of the (e.g., 2×2) grid of pixels or the maximum value of the (e.g., 2×2) grid of pixels. The result of applying a pooling transformation is a downsampled, smaller version of the original image, as shown in FIG. **3** for the transformations of the edge-enhanced image (**304**) by

means of a standard 2×2 pooling operation known in the prior art (**309**) into a downsampled image (**311**) that is 25% the size of the original image (**304**). It will be apparent to those skilled in the art that an attempt to apply standard algorithms for convolution or pooling to a fixed shuffled dataset will not yield the desired result, because the fixed shuffling process has altered the relationship of the pixels in the shuffled image. Therefore, methods are presented in FIG. **3** (and described below, in further detail, with reference to FIG. **4**) for applying modified convolution and pooling algorithms that use the stored lookup table of shuffled indices as described above to apply the modified convolution and modified pooling to preserve the spatial relationships present in the original image.

[0042] Application of a fixed shuffle (**303**) to the starting image (**301**) yields a shuffled image (**305**). Application of a modified convolution algorithm (**307**), in which the original pixel locations are determined by reference to the inverse of the stored fixed shuffle index array, to the shuffled image (**305**) yields a shuffled and convolved image (**308**). Application of a modified pooling algorithm (**310**), in which the original pixel locations are determined by reference to the inverse of the stored fixed shuffle index array, to the shuffled and convolved image (**308**) yields a shuffled, convolved, and pooled image (**312**) that has been downsampled. The convolutional and pooling algorithms applied by reference to the inverse of the stored fixed shuffle array may be applied sequentially, as shown in the illustrative example in FIG. **3**, or separately.

[0043] In alternate embodiments, convolution (**302**) of the unshuffled data (**301**) may be performed first, yielding convolved, unshuffled data (**304**), which may subsequently be subjected to fixed shuffling (**306**) to yield convolved, fixed shuffled data (**308**). Application of a modified pooling algorithm (**310**), in which the original pixel locations are determined by reference to the inverse of the stored fixed shuffle index array, to the shuffled and convolved image (**308**) then yields a shuffled, convolved, and pooled image (**312**) that has been downsampled.

[0044] FIG. **4** illustrates methods to securely utilize shuffled indices and the inverse of shuffled indices to perform functions such as unshuffling, convolution, pooling, or search to fixed shuffled datasets. A set of shuffled indices (**401**) are encrypted (**402**) by standard methods with one or more encryption keys to obtain a set of encrypted shuffled indices (**403**). The set of shuffled indices (**401**) shown is, for purpose of illustration, a short one-dimensional array of 6 values, but within the scope of the invention, the array of shuffled indices may be of any length and of any number of dimensions, and of any data type, suiting the size and shape of the input data arrays. The encrypted shuffled indices (**403**) may be stored and/or transmitted and subsequently decrypted (**404**) by a credentialed user or users with the appropriate key or keys to obtain the set of unencrypted shuffled indices (**405**). The shuffled indices (**405**) can be used to perform a fixed shuffle on the stack of input data (**406**), yielding a stack of fixed shuffled data (**407**). The shuffled indices (**405**) may also be converted to the inverse of the shuffled indices (**409**) by means of a function (**408**) that performs the following operation. The shuffled indices array (**405**) stores the shuffled index values at index positions of an array; the function (**408**) swaps the index values and index positions to yield the inverse of the shuffled indices array (**409**). As the shuffled indices array (**405**) may be used to apply a fixed shuffle to a stack of input data (**406**) to yield a stack of fixed shuffled data (**407**), so may the inverse of shuffled indices array (**409**) be used to unshuffle (**410**) a stack of fixed shuffled data to yield the original stack of data (**406**). The inverse of shuffled indices array acts as a lookup table that maps a value found in a fixed shuffled data array (**407**) back to its position in the original data array (**406**). For example, the position of value '14' shown in the illustrated shuffled dataset (**407**) corresponds to the position of value '0' in in the inverse of shuffled indices array (**409**), mapping the value '14' back to the 0th (initial) index position in the original data array (**406**). Fixed shuffled data (**407**) may be subjected to convolution (**411**) by using the inverse of shuffled indices (**409**) to map values in the fixed shuffled data (**407**) back to their positions in the original data (**406**), yielding convolved shuffled data (**414**). Similarly, fixed shuffled data (**407**) may be subjected to pooling (**412**) by using the inverse of shuffled indices (**409**) to map values in the fixed shuffled data (**407**) back to their positions in the original data (**406**), yielding pooled shuffled data (**415**). Fixed shuffled data (**407**) may be subjected to programmatic search (**413**) by using the inverse of shuffled indices (**409**) to map values in the fixed shuffled data (**407**) back to their positions in the original data (**406**), yielding the locations of any search results in the original data (**416**).

[0045] The inverse of the shuffling index array (**409**) thus functions as a 'lookup table' to locate components of a search in the original data. Thus, for example, the method includes, for each element of a search, locating the element in the padded/shuffled data, then using the inverse of the padding and shuffling index array to determine the location of the element's match in the original data.

[0046] FIG. **5** illustrates methods to securely store and transmit packages of shuffled indices corresponding to the different sizes of data arrays created by pooling. As known in the art and described above, pooling (downsampling) a data array has the effect of reducing the size of the data array. In the example of 2×2 pooling of a two-dimensional data array or image, the data array or image is reduced to 25% of the original size (a 50% reduction in each of the two dimensions). Where it is desirable to apply pooling functions in a neural network used for training on fixed shuffled data, separate fixed shuffled index arrays are generated for each data array or image size (original size and any pooled sizes). For purpose of illustration, an 8×8 data array storing shuffled indices (**501**) is shown. It will be recognized by those skilled in the art that the array of shuffled indices may be of any size and any number of dimensions, to suit the shape of the input data used. A corresponding 4×4 data array storing shuffled indices (**502**) is also shown, as is a corresponding 2×2 data array storing shuffled indices (**503**). The three shuffled index arrays (**501**, **502**, **503**) are used for fixed shuffling of, in this illustrative example, an original data array with shape 4×4×(n examples), a 2×2 pooled version of shape 4×4×(n examples), and a subsequent 2×2 pooled version of shape 2×2×(n examples). The shuffled indices arrays (**501**, **502**, **503**) are collected in a data object (**504**) that in turn is encrypted (**505**) by standard methods to an encrypted data object (**506**) that may be decrypted (**507**) back to the original data object (**504**) for data preparation and use in ANN training and prediction.

[0047] In various embodiments, the data object (**504**) subjected to encryption by standard methods (**505**) also contains corresponding data elements representing one or

more fixed padding elements and one or more fixed perturbation arrays. In various embodiments, fixed padding elements are stored for the original data shape prior to fixed shuffling (as encoded by **501**) along with data stored to indicate the position of the fixed padding element relative to the original data. In other embodiments, fixed padding elements are similarly stored relative to the other levels of fixed shuffling (**502**, **503**). In various embodiments, fixed perturbation arrays are stored for the original data shape prior to fixed shuffling (as encoded by **501**). In other embodiments, fixed perturbation arrays are similarly stored relative to the other levels of fixed shuffling (**502**, **503**).

[0048] In various embodiments, data are prepared for use with ANNs using fixed shuffling alone; in other embodiments, data are prepared for use with ANNs using fixed padding and fixed shuffling; in other embodiments, data are prepared for use with ANNs using fixed perturbation and fixed shuffling; and in other embodiments, data are prepared for use with ANNs using fixed padding, fixed perturbation, and fixed shuffling. In additional embodiments, data are prepared for use with ANNs using various combinations of the three methods of fixed padding, fixed perturbation, and fixed shuffling in various orders, including optional iterative use of two or more rounds of any of the three methods.

[0049] In alternative embodiments, convolutions and pooling are performed prior to fixed padding (if applied), fixed perturbation (if applied), and fixed shuffling. In these alternative embodiments, empiric adjustment of convolutions or pooling during the training process is no longer possible, but in these alternative embodiments, access to fixed shuffling indices and other indices (lookup tables) is not required at training or test time, thereby simplifying workflows.

[0050] In various embodiments, the transformation of data as described in detail above (using various combinations of fixed padding, fixed perturbation, and fixed shuffling) is applied to data for use with other forms of machine learning algorithms beyond the domain of ANNs, including, but not limited to, algorithms supporting supervised machine learning, unsupervised machine learning, and reinforcement machine learning, such as linear regression, polynomial regression, logistic regression, support vector machines, naive Bayes, decision trees, random forests, k-Nearest Neighbors, and ensemble learning approaches.

[0051] FIG. **6** illustrates methods to prepare data for storage and transmission in fixed shuffled packets for use with programmatic search. The programmatic search methods described with reference to FIG. **6** are generally similar to the programmatic search methods described and shown with reference to FIG. **4**. Elements of FIG. **6** having the same name as elements of FIG. **4** are generally the same, except where explicitly stated.

[0052] For illustrative purposes, an original one-dimensional data array (**606**) containing 14 integer values is shown. It will be apparent to those skilled in the art that the data array may have any number of dimensions, with any number of values per dimension, and may store data in integer form, floating point value form, or in the form of another data type. For use with fixed shuffling and programmatic search, the original data array (**606**) is reshaped (**607**) into packets (**608**) of smaller size than the original data array. It will be apparent to those skilled in the art that the packet size may be any size that is smaller than the original data (**606**) size. For purpose of illustration, a packet size of 6 values is shown (**608**), converting the one-dimensional 14-value original dataset (**606**) into a two-dimensional packetized dataset (**608**) of 3 packets of 6 values each, where, for illustrative purposes, zero or Null values are appended to the final packet to bring the length of the final packet to 6 values. To perform fixed shuffling on the packetized dataset (**608**), a shuffled indices array (**601**) of length corresponding to the packet size is subjected to encryption by standard methods (**602**) with one or more encryption keys to yield an encrypted shuffled indices array (**603**) that may be stored or transmitted. The encrypted shuffled indices array (**603**) is decrypted (**604**) with one or more keys to yield the unencrypted shuffled indices array (**605**). The packetized data array (**608**) is subjected to a fixed shuffle (**609**) using the shuffled indices array (**605**), yielding a fixed shuffled packetized data array (**610**). In the illustrative example, each row of the packetized data array (**608**) is subjected to the same shuffle according to the shuffled indices array (**605**), so that a given column of values in the original packetized data array (**608**) is moved together to a new column position in the fixed shuffled packetized data array (**610**). As described with reference to FIG. **4** above, to perform programmatic search, the inverse of the shuffled indices (**612**) is generated by a function (**611**) that swaps the index values and index positions in the shuffled indices array (**605**). Programmatic search (**613**) is then performed by means of the inverse of shuffled indices array (**612**) to yield a search result (**614**) in the original data. For example, using the values shown in the illustration, a search for sequence '14, 86, 21' in the fixed shuffled packetized data (**610**) could first determine the locations of the initial value '14' in the fixed shuffled packetized data (**610**), then convert the locations of these values in the original packetized data (**608**) using the inverse of shuffled indices lookup table (**612**), then look ahead for adjacent values '86' and '21' by the same lookup process. By this process, the location of the search sequence '14, 86, 21' in the fixed shuffled packetized data (**610**) will be determined to start at the 0th (initial) position of the 0th (initial) packet (row) of the original packetized data (**608**), corresponding to the 0th (initial) position in the original data (**606**). The process of looking ahead for adjacent values in this example takes into account the relationship of the end of one packet (row) in the unshuffled packetized data (**608**) with the beginning of the next packet (row) in the unshuffled packetized data (**608**), determined using the lookup via the inverse of the shuffled indices array (**612**), so search sequences crossing packet boundaries (for example, '98, 68, 14') are identified and the location of the search sequence is be determined in the original data (**606**).

[0053] In various embodiments, the shuffled indices (**601**) subjected to encryption by standard methods (**602**) are accompanied prior to encryption by a collection of corresponding data elements representing a fixed padding element and a fixed perturbation array. In various embodiments, a fixed padding element is stored for the original data shape prior to fixed shuffling by the shuffled indices (**601**) along with data stored to indicate the position of the fixed padding element relative to the original data. In various embodiments, a fixed perturbation array is stored for the data shape prior to fixed shuffling by the shuffled indices (**601**).

[0054] In various embodiments, data are prepared for programmatic search using packetization and fixed shuffling alone; in other embodiments, data are prepared for programmatic search using packetization, fixed padding, and fixed

shuffling; in other embodiments, data are prepared for programmatic search using packetization, fixed perturbation, and fixed shuffling; and in other embodiments, data are prepared for programmatic search using packetization, fixed padding, fixed perturbation, and fixed shuffling. In various embodiments, following packetization, the three methods of fixed padding, fixed perturbation, and fixed shuffling are applied in various combinations, and optionally, iterative application of two or more rounds of any of the three methods. In various embodiments, the trailing zeroes shown for purposes of illustration in FIG. **6** as appended to the original data (**606**) to yield the unshuffled packetized data (**608**) with uniform packet size are instead a randomly generated sequence, and the beginning or totality of this randomly generated sequence is stored in encrypted form along with the encrypted shuffled indices to facilitate later separation of original data from the appended random sequence by establishing an end-of-file (EOF) sequence. In further embodiments, the statistical distribution of randomly generated values that are appended to the original data (**606**) to create uniform packet size may be constrained by considerations such as the statistical distribution of values in the parent dataset.

[0055] FIG. **7** illustrates a system in which multiple parties may use secure data for training and implementation of AANs using a computer. Thus, for example and without limitation, an Entity A operates a computer A (**701**), an Entity B operates a computer B (**702**), and a Third Party C operates a server C (**703**). Computer A (**701**), computer B (**702**), and server C (**703**) each including memory and processors that are networked which are programmed, for example and without limitation, to perform the following functions.

[0056] An Entity A has sensitive data stored on computer A (**701**) that it wishes to secure for training in an ANN established by an Entity B on computer (**702**). Server C (**703**) establishes the required fixed padding, fixed perturbation, and fixed shuffling indices (**704**), in accordance with the project data specifications (such as data shape) provided by Entity A and the project training specifications (such as convolutions and/or pooling required) provided by Entity B. The fixed padding, fixed perturbation, and fixed shuffling indices (**704**) are then encrypted and decrypted using standard key pair encryption methods known in the art, as follows. The fixed padding, fixed perturbation, and fixed shuffling indices (**704**) are encrypted (**705**) with a Key C1 (**706**) belonging to Third Party C, then encrypted (**707**) with a Key A1 (**708**) belonging to Entity A that is passed from computer A (**701**) to server (**703**), yielding an encrypted version of the fixed padding, fixed perturbation, and fixed shuffling indices (**709**). A copy of the encrypted fixed padding, fixed perturbation, and fixed shuffling indices (**710**) is passed from server C (**703**) to computer A (**701**), where a decryption step (**711**) is performed with the second component of a key pair, Key A2 (**712**), belonging to Entity A. The result of this first decryption step is passed to a local client software (**713**) that was previously established by Third Party C, and which is running behind the firewall of computer A. Within the local client software C (**713**), a second decryption step (**714**) is performed using the second component of a key pair, Key C2 (**715**), belonging to Third Party C and stored in the local client software (**713**), yielding a decrypted version of the fixed padding, fixed perturbation, and fixed shuffling indices (**716**) that corresponds to the original fixed padding, fixed perturbation, and fixed shuffling indices (**704**) generated in the server belonging to Third Party C. Training data (**717**) stored on computer A (**703**) is then passed to the local client software (**713**) and transformed by means of the fixed padding, fixed perturbation, and fixed shuffling indices (**716**) to yield fixed padded, fixed perturbed, and fixed shuffled training data (**718**) that is passed back to computer A (**701**). The fixed padded, fixed perturbed, and fixed shuffled training data (**718**) are then transmitted to computer B (**702**) for training of an ANN (**718**) to yield a trained neural network (**719**).

[0057] In alternate embodiments, the fixed padded, fixed perturbed, and fixed shuffled training data (**718**) are transmitted directly from the local client software (**713**) to the neural network (**718**) running in software controlled by Entity B on computer B (**702**). In other alternate embodiments in which Entity A is not required to be part of the encryption and decryption process for the encrypted fixed padding, fixed perturbation, and fixed shuffling indices (**709**), standard public/private key pair encryption such as RSA are applied to securely transmit the encrypted fixed padding, fixed perturbation, and fixed shuffling indices (**709**) between the server C (**703**) and the local client software (**713**).

[0058] For use of the trained neural network to analyze test data, Software D (**720**) is generated that takes test data (**721**) and transforms the data into fixed padded, fixed perturbed, and fixed shuffled test data (**724**) using either client software C (**722**) or server software C (**723**) to transform the data as described above. When this transformation takes place, this event is recorded (**725**) into a ledger maintained in server software C (**726**), and both Entity A, via computer A (**701**) and Entity B, via computer B (**702**) have access to this ledger, by means of permissioned access to a private file maintained in server software C (**726**) or by alternative technologies known in the art, such as permissioned blockchain. The fixed padded, fixed perturbed, and fixed shuffled test data (**724**) is then processed using a copy of the trained neural network (**727**) passed from computer B (**702**) to Software D (**720**), to yield a result (**728**). In alternate embodiments, the yielding of the result (**728**) from the trained neural network (**727**) is recorded in the ledger maintained in server software C (**726**), in place of, or in addition to, the recordation of the data transformation event (**725**).

[0059] FIG. **8** illustrates systems wherein files on a computer system or cloud service may be stored securely in a format that allows for search and file restoration by credentialed users. The system components include memory and processors that are networked which are programmed, for example and without limitation, to perform the following programmatic search methods illustrated in FIG. **8** are generally similar to the programmatic search methods described and shown with reference to FIGS. **4** and/or **6**. Elements of FIG. **8** having the same name as elements of FIGS. **4** and/or **6** are generally the same, except where explicitly stated.

[0060] On a computer system or cloud service (**801**), an unencrypted memory storage (**802**) contains any number of files (**803**) stored in memory in conventional unencrypted form. As depicted for purposes of graphical illustration by the height of the rectangles representing individual data files (**803**), the unencrypted files may be of arbitrary size. An initial transformation step is performed on the files (**803**) in

memory storage (**802**) to subdivide a copy of the files (**803**) into packetized form (**805**) in memory storage (**804**), with packets of uniform size as illustrated graphically (**805**). The uniform file packets (**805**) in memory storage (**804**) are then transformed into data arrays (**807**) with the shape determined by ([packet size]×[number of packets]) in memory storage (**806**). In a preferred embodiment, fixed padding data and fixed shuffle indices (**808**) are generated that correspond to the data shape of the arrays (**807**), in the fashion described in detail above, and stored temporarily for use in transforming the data arrays (**807**) in memory storage (**806**) into fixed padded and fixed shuffled data arrays (**809**). In addition, the fixed padding and fixed shuffling indices (**808**) are encrypted (**810**) using a first key of a key pair, Key 1 (**811**) to yield encrypted fixed padding and fixed shuffling indices (**812**). After generation of the encrypted fixed padding and fixed shuffling indices (**812**), the original fixed padding and fixed shuffling indices (**808**) are deleted from memory.

[0061] When a search query (**815**) is produced by a credentialed user, a second key of a key pair, Key 2 (**814**) is used to decrypt (**813**) the encrypted fixed padding and fixed shuffling indices (**812**) to yield a decrypted version of the fixed padding and fixed shuffling indices (**816**). The fixed padding and fixed shuffling indices (**816**) are then applied to the search query and the result is used to perform a search (**817**) on the fixed padded and fixed shuffled data arrays (**809**) to yield a search result in the original data (**818**) using the method described in detail above. After the search is performed, the decrypted fixed padding and fixed shuffling indices (**816**) are deleted from memory.

[0062] When a file request (**819**) is produced by a credentialed user, a second key of a key pair, Key 2 (**814**) is used to decrypt (**813**) the encrypted fixed padding and fixed shuffling indices (**812**) to yield a decrypted version of the fixed padding and fixed shuffling indices (**816**). The file request (**819**) is then addressed using the decrypted fixed padding and fixed shuffling indices (**816**) to restore the requested file (**820**) from the fixed padded and fixed shuffled data arrays (**809**), to yield an unpadded and unshuffled data array for the requested file (**821**). The unpadded and unshuffled data array for the requested file (**821**) is then transformed into the corresponding data packets of fixed size (**822**), which are then rejoined to yield the original file (**823**). After restoration of the original file (**823**), the decrypted fixed padding and fixed shuffling indices (**816**) are deleted from memory. In alternative embodiments for implementation of secure search and file restoration, fixed perturbation is performed between the fixed padding and fixed shuffling, as described in detail above.

[0063] FIG. **9** illustrates systems wherein files, messages, posts, notes, or other electronic communications on a mobile, online, Web-based, or cloud-based electronic communication system or computer system or cloud service may be stored securely in a format that allows for keyword matching search and secure neural network training by permissioned entities. The components of FIG. **9** include memory and processors that are networked which are programmed, for example and without limitation, to functions described herein. It will be apparent to those skilled in the art that the disclosed invention to secure data in searchable form and the disclosed invention to secure data for training neural networks, alone or in combination, yield certain advantageous embodiments of potential commercial interest. In one such embodiment, a mobile, online, Web-based, or cloud-based electronic communication system or other storage medium (**901**) contains a corpus of any number of data files, messages, posts, notes, or other electronic communications (**902**), and these are converted according to the methods disclosed above into a fixed padded and fixed shuffled corpus (**903**). Where it is desirable to obtain keyword matching for third-party content (including, but not limited to, advertising content), permissions (**905**) are obtained, and keyword search (**906**) is performed according to the methods disclosed above. The use of the term 'keyword' as used here includes any search for a content match of interest in the secured data contained in the fixed padded and fixed shuffled corpus (**903**), including any data pattern of interest in data type stored therein, including binary or machine representations, text, images, video, or other forms of matchable digital content. In similar fashion, keyword matching may be performed for other purposes, including, but not limited to, content organization within the corpus of information (**907**). Because the fixed padded and fixed shuffled corpus of information (**903**) is organized into data arrays of fixed shape, as disclosed above, the fixed-shape packets of fixed padded and fixed shuffled data (**903**) may be used to train a neural network (**908**) to yield a trained neural network (**909**). In certain embodiments, the neural network (**908**) implemented in this context represents unsupervised machine learning to organize the data contained in the fixed padded and fixed shuffled corpus of information (**903**). In other embodiments, the neural network (**908**) implemented in this context represents supervised machine learning using labels obtained from metadata, user-provided data, the results of programmatic keyword matching (**904**, **907**), or other data sources of interest. In other embodiments, the neural network (**908**) implemented in this context represents other types of machine learning approaches known in the art, including but not limited to semi supervised learning, dimensionality reduction, anomaly detection, or reinforcement learning. As described above for the other machine learning processes disclosed in the present invention, the neural network (**908**) implemented in this context may be, in alternative embodiments, another form of machine learning algorithms outside the domain of ANNs, including, but not limited to, algorithms supporting supervised machine learning, unsupervised machine learning, and reinforcement machine learning, such as linear regression, polynomial regression, logistic regression, support vector machines, naive Bayes, decision trees, random forests, k-Nearest Neighbors, and ensemble learning approaches.

[0064] The present invention is not intended to be limited to a system or method which must satisfy one or more of any stated or implied objects or features of the invention. Modifications and substitutions by one of ordinary skill in the art are considered to be within the scope of the present invention. Numerous details are provided to convey an understanding of the embodiments described herein. It will be understood by those of ordinary skill in the art that the embodiments described herein may be practiced without these specific details. In other instances, well-known methods, procedures, and components have not been described in detail so as not to obscure the embodiments described herein. The present invention is not limited to the preferred, exemplary, or primary embodiment or embodiments described herein.

[0065] In addition, while the invention has been described in terms of a number of different functions or steps, it will

be appreciated by those skilled in the art that the order of the functions or steps may be performed in a different order than as described herein, and that certain functions or steps may be combined into to a fewer number or greater number of steps to achieve the same effect as is described herein.

[0066] It will also be understood by one of ordinary skill in the art that the systems and methods may be provided on many different types of computer-readable media including computer storage mechanisms that contain instructions for use in execution by a processor to perform the methods' operations and implement the systems described herein. Any unit, component, computer, module, server, terminal, or device described or exemplified herein that executes instructions may include or otherwise have access to computer readable media such as storage media, computer storage media, or data storage devices (removable and/or non-removable, volatile and/or non-volatile) such as, for example, CD-ROM, diskette, RAM, ROM, EEPROM, flash memory, computer hard drive, magnetic disks, optical disks, tape, or other memory technology implemented in any method for storage or transmission of information, such as computer readable instructions, data structures, program modules, or other data. Any such computer storage media may be part of the device or accessible or connectable thereto. Any application or module herein described may be implemented using computer readable and/or executable instructions that may be stored or otherwise held by such computer-readable media.

[0067] With respect to the appended claims, unless stated otherwise, the term "first" does not, by itself, require that there also be a "second." Moreover, reference to only "a first" and "a second" does not exclude additional items. While the particular computer-based systems and methods described herein and described in detail are fully capable of attaining the above-described objects and advantages of the invention, it is to be understood that these are the presently preferred embodiments of the invention and are thus representative of the subject matter which is broadly contemplated by the present invention, that the scope of the present invention fully encompasses other embodiments which may become obvious to those skilled in the art, and that the scope of the present invention is accordingly to be limited by nothing other than the appended claims, in which reference to an element in the singular means "one or more" and not "one and only one," unless otherwise so recited in the claim.

[0068] Although the invention has been described relative to specific embodiments thereof, there are numerous variations and modifications that will be readily apparent to those skilled in the art in light of the teachings presented herein.

1. A method for using modified data with a neural network, said method comprising:

determining an algorithm to modify a plurality of examples, where each example of the plurality of examples includes an array of values, and where each example is a training example or a test example;

training the neural network by:

obtaining a plurality of training examples,

modifying each training example of said plurality of training examples according to said algorithm to form a plurality of modified training examples, and

forming a trained neural network by training a neural network with said plurality of modified training examples; and

forming predictions using the trained neural network by:

accepting a test example,

modifying the test example according to said algorithm to form a modified test example, and

forming a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input, such that the same algorithm modifies each training example and the test example.

2. The method of claim **1**, further comprising:

prior to forming predictions using the trained neural network, encrypting said algorithm to form an encrypted algorithm, and

where said forming predictions using the trained neural network further includes accepting said encrypted algorithm and decrypting said encrypted algorithm.

3. The method of claim **1**,

where said algorithm includes a first pad of values,

where said modifying each training example includes increasing the size of each training example by appending the first pad of values to each training example, and

where said modifying the test example includes increasing the size of the test example by appending the first pad of values to the test example,

such that the same pad of values is applied to each training example and the test example.

4. The method of claim **1**,

where said algorithm includes a plurality of perturbation functions, where each perturbation function of the plurality of perturbation functions corresponds to a position in the array of each training example and to a position in the array of the test example,

where said modifying each training example includes is the mathematical equivalent of applying each perturbation function to the value in the corresponding position in the training example, and

where said modifying the test example is the mathematical equivalent of applying each perturbation function to the value in the corresponding position in the test example,

such that the same perturbation function is applied to corresponding array values of each training example and of the test example.

5. The method of claim **1**,

where said algorithm includes an index shuffling,

where said modifying each training example includes applying the index shuffling to the training example, and

where said modifying the test example includes applying the index shuffling to the test example,

such that the same index shuffling is applied to each training example and to the test example.

6. The method of claim **1**, where said algorithm is the mathematical equivalent to two or more modifications performed sequentially, where the two or more modifications include two or more of:

a) one or more paddings each including a pad of values, where said modifying each training example and said modifying the test example includes appending the pad of values to each example or previously modified example;

b) one or more perturbations each including an array of perturbation functions, where each perturbation function corresponds to a position in the array of the training example and to a position in the array of the

test example, where said modifying each training example and modifying the test example applies each perturbation function to the value in the corresponding position in each example or previously modified example; and

c) one or more index shuffles each including an index shuffling for each index shuffle, where said modifying each training example and said modifying the test example includes applying the index shuffling to each example or previously modified example.

7. The method of claim **1**, where said forming the trained neural network includes using a mathematical representation of the modifying of each training example.

8. The method of claim **1**, where said steps of said training the neural network are performed by two or more parties.

9. The method of claim **1**, where said steps of said forming predictions using the trained neural network are performed by two or more parties.

10. The method of claim **1**, further including storing, in a ledger, the occurrence of said modifying a test example or the occurrence of said providing said trained neural network with a modified test example to form a prediction.

11. A system for using modified data with a neural network, said system including networked memory and processors programmed to:

determine an algorithm to modify a plurality of examples, where each example of the plurality of examples includes an array of values, and where each example is a training example or a test example;

train the neural network by the processors programmed to:

obtain a plurality of training examples,

modify each training example of said plurality of training examples according to said algorithm to form a plurality of modified training examples, and

form a trained neural network by training a neural network with said plurality of modified training examples; and

form predictions using the trained neural network by the processors programmed to:

accept a test example,

modify the test example according to said algorithm to form a modified test example, and

form a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input,

such that the same algorithm is used to modify each training example and the test example.

12. The system of claim **11**, where said processors are further programmed to prior to said form predictions, encrypt said algorithm to form an encrypted algorithm, and

where said form predictions further includes accept said encrypted algorithm and where said processors are further programmed to decrypt said encrypted algorithm.

13. The system of claim **11**, where said algorithm is the mathematical equivalent to one or more modifications performed sequentially, where the one or more modifications include one or more of:

a) one or more paddings each including a pad of values, where said modify each training example and said modify the test example includes said processors further programmed to append the pad of values to each example or previously modified example;

b) one or more perturbations each including an array of perturbation functions, where each perturbation function corresponds to a position in the array of the training example and to a position in the array of the test example, where said modify each training example and said modify the test example includes said processors further programmed to apply each perturbation function to the value in the corresponding position in each example or previously modified example; and

c) one or more index shuffles each including an index shuffling for each index shuffle, where said modify each training example and said modify the test example includes said processors further programmed to apply the index shuffling to each example or previously modified example.

14. The system of claim **11**, where said form the trained neural network includes said processors further programmed to use a mathematical representation of the modifying of each training example.

15. The system of claim **11**, where said processors are further programmed to store, in a ledger, the occurrence of said modify a test example or the occurrence of said provide said trained neural network with a modified test example to form a prediction.

16. A method for using modified data with a neural network, where the neural network is trained using a plurality of modified training examples, where each training example of the plurality of training examples is modified using an algorithm, said method comprising:

accepting a test example,

accepting the algorithm used for modifying each training example of the plurality of training examples,

modifying the test example using the algorithm, and

forming a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input,

such that the same algorithm is used to modify each training example and the test example.

17. The method of claim **16**, where the algorithm is an encrypted algorithm, and where said accepting the algorithm includes accepting the encrypted algorithm and decrypting said encrypted algorithm.

18. The method of claim **16**,

where the algorithm includes a first pad of values, and

where said modifying the test example includes increasing the size of the test example by appending the first pad of values to the test example,

such that the same pad of values is applied to each training example and the test example.

19. The method of claim **16**,

where the algorithm includes a plurality of perturbation functions, where each perturbation function of the plurality of perturbation functions corresponds to a position in the array of the test example, and

where said modifying the test example is the mathematical equivalent of applying each perturbation function to the value in the corresponding position in the test example,

such that the same perturbation function is applied to corresponding array values of each training example and of the test example.

20. The method of claim **16**,
where the algorithm includes an index shuffling, and
where said modifying the test example includes applying the index shuffling to the test example,
such that the same index shuffling is applied to each training example and to the test example.

21. The method of claim **16**, where said algorithm is the mathematical equivalent to two or more modifications performed sequentially, where the two or more modifications include two or more of:
a) one or more paddings each including a pad of values, where said modifying the test example includes appending the pad of values to the test example or to a previously modified test example,
b) one or more perturbations each including an array of perturbation functions, where each perturbation function corresponds to a position in the array of the test example, where said modifying applies each perturbation function to the value in the corresponding position in the test example or to a previously modified test example, and
c) one or more index shuffles each including an index shuffling, where said modifying the test example includes applying the index shuffling to the test example or to a previously modified test example.

22. The method of claim **16**, where said forming the trained neural network includes using a mathematical representation of the modifying of each training example.

23. The method of claim **16**, further including recording, in a ledger, the occurrence of said modifying each test example or the occurrence of said providing said trained neural network with one or more modified test examples.

24. A system for using modified data with a neural network, where the neural network is trained using a plurality of modified training examples, where each training example of the plurality of training examples is modified using an algorithm, said system including networked memory and processors programmed to:
accept a test example,
accept the algorithm used to modify each training example of the plurality of training examples,
modify the test example using the algorithm, and
form a prediction from the output of the trained neural network by providing said trained neural network with the modified test example as input,
such that the same algorithm is used to modify each training example and the test example.

25. The system of claim **24**, where the algorithm is an encrypted algorithm, and where said processor is further programmed to accept the algorithm by decrypting said encrypted algorithm.

26. The system of claim **24**, where said algorithm is the mathematical equivalent to one or more modifications performed sequentially, where the one or more modifications include one or more of:
a) one or more paddings each including a pad of values, where said modify the test example includes said processors further programmed to append the pad of values to each test example or previously modified test example;
b) one or more perturbations each including an array of perturbation functions, where each perturbation function corresponds to a position in the array of the test example, where said modify the test example includes said processors further programmed to apply each perturbation function to the value in the corresponding position in each test example or previously modified test example; and
c) one or more index shuffles each including an index shuffling for each index shuffle, where said modify the test example includes said processors further programmed to apply the index shuffling to each test example or previously modified test example.

27. The system of claim **24**, where said form the trained neural network includes using a mathematical representation of the modifying of each training example.

28. The system of claim **24**, where said processors are further programmed to store, in a ledger, the occurrence of said modify a test example or the occurrence of said provide said trained neural network with a modified test example to form a prediction.

* * * * *