(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 October 2022 (13.10.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/216591 A1

(51) International Patent Classification:
G16B 30/00 (2019.01) *G06N 3/08* (2006.01)
G16B 40/00 (2019.01) *G06N 3/04* (2006.01)

(21) International Application Number:
PCT/US2022/023280

(22) International Filing Date:
04 April 2022 (04.04.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/170,697 05 April 2021 (05.04.2021) US
17/711,617 01 April 2022 (01.04.2022) US

(71) Applicant: **NEC LABORATORIES AMERICA, INC.** [US/US]; 4 Independence Way, Suite 200, Princeton, New Jersey 08540 (US).

(72) Inventors: **MIN, Renqiang**; 7 Carlton Circle, Princeton, New Jersey 08540 (US). **GRAF, Hans Peter**; 55 South Shore Drive, South Amboy, New Jersey 08879 (US). **HAN, Ligong**; 61 Bartha Avenue, Edison, New Jersey 08817 (US).

(74) Agent: **BITETTO, James J.**; 401 Broadhollow Road, Suite 402, Melville, New York 11747 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,

(54) Title: GENERATING MINORITY-CLASS EXAMPLES FOR TRAINING DATA

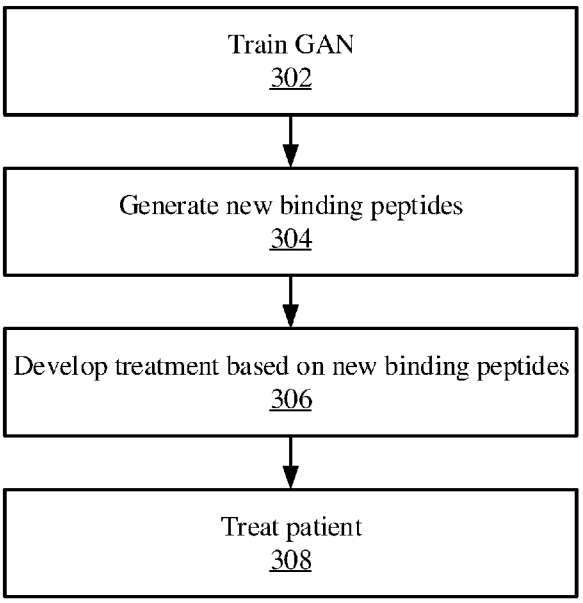


FIG. 3

(57) Abstract: Methods and systems for training a model include encoding (203) training peptide sequences using an encoder model. A new peptide sequence is generated (202) using a generator model. The encoder model, the generator model, and the discriminator model are trained (206) to cause the generator model to generate new peptides that the discriminator mistakes for the training peptide sequences, including learning projection vectors with respective cross-entropy losses for binding sequences and non-binding sequences.

WO 2022/216591 A1

[Continued on next page]

TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

GENERATING MINORITY-CLASS EXAMPLES FOR TRAINING DATA

RELATED APPLICATION INFORMATION

[0001] This application claims priority to U.S. Non-Provisional Patent Application No. 17/711,617, filed on April 1, 2022 and U.S. Provisional Patent Application No. 63/170,697, filed on April 5, 2021, both incorporated herein by reference in its entirety.

BACKGROUND

Technical Field

[0002] The present invention relates to neural network training, and, more particularly, to generating minority-class examples for enhancing neural network training data.

Description of the Related Art

[0003] Peptide-MHC (Major Histocompatibility Complex) protein interactions are involved in cell-mediated immunity, regulation of immune responses, and transplant rejection. While computational tools exist to predict a binding interaction score between an MHC protein and a given peptide, tools for generating new binding peptides with new specified properties from existing binding peptides are lacking.

SUMMARY

[0004] A method for training a model includes encoding training peptide sequences using an encoder model. A new peptide sequence is generated using a generator model. The encoder model, the generator model, and the discriminator model are trained to cause the generator model to generate new peptides that the discriminator mistakes for

the training peptide sequences, including learning projection vectors with respective cross-entropy losses for binding sequences and non-binding sequences.

[0005] A method for developing treatments includes training a generative adversarial network (GAN) model to generate binding peptide sequences relating to a major histocompatibility complex (MHC) protein associated with a virus pathogen or tumor. A new binding peptide sequence is generated using the trained GAN. A treatment for the virus pathogen or tumor is developed associated with the MHC protein using the new binding peptide sequence.

[0006] A system for training a model includes a hardware processor and a memory that stores a computer program. When executed by the hardware processor, the computer program causes the hardware processor to encode training peptide sequences using an encoder model, to generate a new peptide sequence using a generator model, and to train the encoder model, the generator model, and the discriminator model to cause the generator model to generate new peptides that the discriminator mistakes for the training peptide sequences, including learning projection vectors with respective cross-entropy losses for binding sequences and non-binding sequences.

[0007] These and other features and advantages will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0008] The disclosure will provide details in the following description of preferred embodiments with reference to the following figures wherein:

[0009] FIG. 1 is a diagram that illustrates binding between a peptide and a major histocompatibility complex (MHC), in accordance with an embodiment of the present principles;

[0010] FIG. 2 is a block diagram of a generative adversarial network (GAN) that can be trained to generate binding peptide sequences, in accordance with an embodiment of the present invention;

[0011] FIG. 3 is a block/flow diagram of a method for developing and administering a treatment for a given pathogen, in accordance with an embodiment of the present invention;

[0012] FIG. 4 is a block/flow diagram of a method for training a GAN to generate peptide sequences that can bind to a given MHC protein, in accordance with an embodiment of the present invention;

[0013] FIG. 5 is a block diagram of a neural network architecture of an exemplary peptide sequence discriminator, in accordance with an embodiment of the present invention;

[0014] FIG. 6 is a block diagram of a neural network architecture of an exemplary peptide sequence classifier, in accordance with an embodiment of the present invention;

[0015] FIG. 7 is a block diagram of a neural network architecture of an exemplary peptide sequence generator, in accordance with an embodiment of the present invention;

[0016] FIG. 8 is a diagram of a patient being treated using a treatment developed by generating a new binding peptide for a specific major histocompatibility complex, in accordance with an embodiment of the present invention;

[0017] FIG. 9 is a block diagram of a computing device that includes program code for training a model and generating new binding peptide sequences, in accordance with an embodiment of the present invention;

[0018] FIG. 10 is a diagram of an exemplary neural network architecture that may be used to implement one or more models, in accordance with an embodiment of the present invention; and

[0019] FIG. 11 is a diagram of an exemplary neural network architecture that may be used to implement one or more models, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0020] Protein interactions between peptides and major histocompatibility complexes (MHCs) are involved in cell-mediated immunity, regulation of immune responses, and transplant rejection. Machine learning systems, including regression-based methods and neural network–based methos, may generate a prediction for a binding interaction score between an MHC protein and a given peptide. A machine learning system, as described herein, may generate new peptides with a strong binding interaction score with the MHC protein, based on one or more starting peptides.

[0021] Such generative systems may assume that the provided binding peptides are sufficient to train a generative model, such as a conditional generative adversarial network (GAN). However, new binding peptides may be generated, even when the provided training dataset is imbalanced, with a number of binding peptides being significantly smaller than the number of non-binding peptides.

[0022] The training dataset may be enhanced by introducing additional minority-class training examples. While the specific application to generating binding peptides

is described in detail herein, it should be understood that the training dataset enhancement described herein may be applied to a variety of different applications where training data for a category to be identified may be scarce, such as in visual product defect classification and anomaly detection.

[0023] New binding peptides may be generated using a deep generative system that is trained using a dataset with both MHC-binding peptides and non-binding peptides. Instead of predicting binding scores of a predefined set of peptides, the conditional GAN is trained on MHC-binding peptides with dual class label projections and a generator with tempering softmax units.

[0024] A conditional Wasserstein GAN may be trained using a dataset that includes both binding and non-binding peptide sequences for an MHC. The conditional Wasserstein GAN may include a generator and a discriminator, with the generator being a deep neural network that transforms a sampled latent code vector z and a sampled label y to a generated peptide sequence.

[0025] Referring now to FIG. 1, a diagram of a peptide-MHC protein bond is shown. A peptide 102 is shown as bonding with an MHC protein 104, with complementary two-dimensional interfaces of the figure suggesting complementary shapes of these three-dimensional structures. The MHC protein 104 may be attached to a cell surface 106.

[0026] An MHC is an area on a DNA strand that codes for cell surface proteins that are used by the immune system. MHC molecules are used by the immune system and contribute to the interactions of white blood cells with other cells. For example, MHC proteins impact organ compatibility when performing transplants and are also important to vaccine creation.

[0027] A peptide, meanwhile, may be a portion of a protein. When a pathogen presents peptides that are recognized by a MHC protein, the immune system triggers a response to destroy the pathogen. Thus, by finding peptide structures that bind with MHC proteins, an immune response may be intentionally triggered, without introducing the pathogen itself to a body. In particular, given an existing peptide that binds well with the MHC protein 104, a new peptide 102 may be automatically identified according to desired properties and attributes.

[0028] Although the present principles are described with specific focus on the generation of binding peptides, they may be readily extended to include continuous binding affinity predictions of peptide sequences, naturally processed peptide predictions of peptide sequences, T-cell epitope predictions of peptide sequences, etc. Varying the application involves providing different supervision signals for optimizing the cross-entropy loss terms, described in greater detail below.

[0029] Furthermore, the present principles are not limited to binding peptide generation, but may be extended to generate other minority-class examples with other applications. For example, minority-class product images may be generated for product inspection and anomaly detection. For such tasks, the input training data may include images, and the generator architecture may be altered to accommodate that input format.

[0030] Referring now to FIG. 2, an exemplary GAN 200 is shown. The GAN 200 includes a generator 202 and a discriminator 204. The generator 202 generates training dataset candidates, while the discriminator 204 attempts to distinguish between the generated candidates and true samples from a provided training dataset 201. An encoder 203 converts the sequences of the training dataset into vectors in an embedded space. The encoder may use block substitution or a pre-trained amino acid embedding scheme

to convert the amino acid sequence into, e.g., a feature representation matrix, with each column of the matrix corresponding to an amino acid. The encoder 203 and the generator 202 may be trained together to fool the discriminator.

[0031] The generator 202 is trained to increase the error rate of the discriminator 204, while the discriminator 204 is trained to decrease its error rate in identifying the generated candidates. A trainer 206 uses a loss function to perform training for the generator 202 and the discriminator 204. In a Wasserstein GAN, the loss function may be based on the Wasserstein metric.

[0032] In the context of peptide generation, the training dataset 201 may include both binding and nonbinding peptide sequences that interact with an MHC. The generator 202 may be a deep neural network, which transforms a sampled latent code vector z from a multivariate unit-variance Gaussian distribution and a sampled binding class label (e.g., 1 for "binding" and 0 for "non-binding") to a peptide feature representation matrix, with each column corresponding to an amino acid.

[0033] The discriminator 204 may be a deep neural network with convolutional layers and fully connected layers between an input representation layer and an output layer that outputs a scalar value. The parameters of the discriminator 204 may be updated to distinguish generated peptide sequences from sampled peptide sequences in the training dataset 201. The parameters of the generator 202 are updated to fool the discriminator 204.

[0034] A dual-projection GAN can be used to simultaneously learn two projection vectors, with two cross-entropy losses for each class (e.g., "binding" and "non-binding"). This is equivalent to maximizing the mutual information between generated data examples and their associated labels, with one loss discriminating between real binding/non-binding peptides in the training data and real non-binding/binding peptides

in the training data, and the other loss discriminating between generating binding/non-binding peptides and generated non-binding/binding peptides. The generator 202 may be updated to minimize these two cross-entropy losses for each class.

[0035] A non-negative scalar weight $\lambda(x)$ may be learned for each data point x associated with the two cross-entropy losses, balancing the discriminator loss. A penalty term of $-0.5\log(\lambda(x))$ may be added to penalize large values of $\lambda(x)$. Data-label pairs may be denoted as $\{x_i, y_i\}_{i=1}^{n} \subseteq \mathcal{X} \times \mathcal{Y}$, drawn from a joint distribution P_{XY}, where x is a peptide sequence and y is a label. The generator 202 is trained to transform samples $z \sim P_Z$ from a canonical distribution conditioned on labels to match the real data distributions, with real distributions being denoted as P and with generated distributions being denoted as Q. The discriminator 204 learns to distinguish samples drawn from the joint distribution P_{XY} and Q_{XY}.

[0036] Discriminator and generator loss terms may be written as the following objectives:

$$L_D = \mathbb{E}_{x,y\sim P_{XY}}\mathcal{A}\left(-\widetilde{D}(x,y)\right) + \mathbb{E}_{z\sim P_Z, y\sim Q_Y}\mathcal{A}\left(\widetilde{D}(G(z,y),y)\right)$$

$$L_G = \mathbb{E}_{z\sim P_Z, y\sim Q_Y}\mathcal{A}\left(-\widetilde{D}(G(z,y),y)\right)$$

where $\mathcal{A}(\cdot)$ is an activation and $\widetilde{D}$ is the discriminator's output before activation. The activation function may be $\mathcal{A}(t) = softplus(t) = \log(1+e^t)$. With this activation function, the logit of an optimal discriminator can be decomposed in two ways:

$$\widetilde{D}^*(x,y) = \log\left(\frac{P(x)}{Q(x)}\right) + \log\left(\frac{P(y|x)}{Q(y|x)}\right)$$

$$\widetilde{D}^*(x,y) = \log\left(\frac{P(x|y)}{Q(x|y)}\right) + \log\left(\frac{P(y)}{Q(y)}\right)$$

[0037] The logic of a projection discriminator can be derived as:

$$\widetilde{D}(x,y) = v_y^{\mathrm{T}}\phi(x) + \psi\big(\phi(x)\big)$$

where $\phi(\cdot)$ is the image embedding function, v_y is an embedding of class y, and ψ collects residual terms. The term v_y can be expressed as a difference of real and generated class embeddings, $v_y = v_y^p - v_y^q$.

[0038] Thus, a projection discriminator can tie the parameters v_y^p and v_y^q to a single v_y. Tying embeddings can turn the problem of learning categorical decision boundaries into learning a relative translation vector for each class, which is a simpler process. Without loss of generality, the term $\psi(\cdot)$ may be assumed to be a linear function v_ψ. The softplus function may be approximated by $ReLU = \max(0,\cdot)$, which produces a large loss when x^+ and x^- are misclassified. Thus, learning can be performed by alternating the steps:

Discriminator: Align $\big(v_y + v_\psi\big)$ with $\big(\phi(x^+) - \phi(x^-)\big)$

Generator: Move $\phi(x^-)$ along $\big(v_y + v_\psi\big)$

By tying the parameters, the GAN can directly perform data matching without explicitly enforcing label matching, aligning $Q(x|y)$ with $P(x|y)$.

[0039] The term v_y should recover the difference between the underlying v_y^p and v_y^q, but to explicitly enforce that property, the class embeddings may be separated out, and V^p and V^q may be used to learn conditional distributions $p(y|x)$ and $q(y|x)$, respectively. This may be done with the softmax function, and cross-entropy losses may be expressed as:

$$L_{mi}^p = -v_y^{p\mathrm{T}}\phi(x^+) + \log\sum_{y'} e^{v_{y'}^{p\mathrm{T}}\phi(x^+)}$$

$$L_{mi}^{q} = -v_{y}^{q\mathrm{T}}\phi(x^{-}) + \log\sum_{y'} e^{v_{y'}^{q\mathrm{T}}\phi(x^{-})}$$

$$L_{D}^{P2} = L_{D}(\widetilde{D}) + L_{mi}^{p} + L_{mi}^{q}$$

$$L_{G}^{P2} = L_{G}(\widetilde{D})$$

$$\widetilde{D} = \left(v_{u}^{p} - v_{u}^{q}\right)^{\mathrm{T}}\phi(x) + \psi\left(\phi(x)\right)$$

where p and q correspond to conditional distribution or loss function using real/generated binding peptides, the terms v_{y}^{p} and v_{y}^{q} represent embeddings of the real and generated samples, respectively, $\phi(\cdot)$ is an embedding function, $\psi(\cdot)$collects residual terms, and $x^{+}\sim P_{X}$ and $x^{-}\sim Q_{X}$ are real and generated sequences (with P and Q being the respective real and generated distributions), and y is a data label. The classifiers V^{p} and V^{q} are trained on real data and generated data, respectively. The discriminator loss L_{D}^{P2} and generator loss L_{G}^{P2} trained as above. Both $L_{D}(\widetilde{D})$ and L_{mi}^{p} include the parameter V^{p}, while $L_{D}(\widetilde{D})$ and L_{mi}^{q} both include V^{q}.

[0040] Data matching and label matching may be weighted by the model. A gate may be added between the two losses:

$$L_{D}^{P2w} = L_{D} + \lambda\left(L_{mi}^{p} + L_{mi}^{q}\right)$$

The definition of λ changes the behavior of the system. Variants may include exponential decay, scalar valued, and amortized models. For example, λ may be defined as a decaying factor, $\lambda = e^{-\frac{t}{T}}$, where t is a training iteration and T is a maximum number of training iterations.

[0041] In a scalar valued embodiment, if $\lambda \geq 0$ is a learnable parameter, initialized as 1, class separation may be enforced as long as $\lambda > 0$. A penalty term may be used:

$$L_{D}^{P2sp} = L_{D} + \lambda\left(L_{mi}^{p} + L_{mi}^{q}\right) - \frac{1}{2}\log\lambda$$

[0042] In an amortized embodiment, amortized homoscedastic weights may be learned for each data point. The term $\lambda(x) \geq 0$ would then be a function of x producing per-sample weights. A penalty can be added. When loss terms involve non-linearity in the mini-batch expectation, any type of linearization may be applied.

[0043] Softmax may be used in the last output layer of the generator 202, with entropy regularization being used to implicitly control the temperature in the tempering softmax units. In a forward pass, a straight-through estimator may be used to output discrete amino acid sequences (e.g., peptides) with "binding" or "non-binding" labels. In the backward pass, the temperatures may be used to facilitate continuation gradient calculations. At the beginning of training, a smaller penalty coefficient may be set for entropy regularization to encourage more uniform amino acid emission probability distributions. Later in training, a larger penalty coefficient may be used for entropy regularization to encourage amino acid emission probability distributions with more peaks.

[0044] Besides updating the discriminator 204 and generator 202 in a weighted framework, an encoder may be trained to map an input peptide sequence x to a latent embedding code space z. The aggregated latent codes of the input peptide sequences may be enforced to follow a multivariate unit-variance Gaussian distribution, by minimizing a kernel maximum mean discrepancy regularization term. Each embedding code z is fed into the generator 202 to reconstruct the original peptide sequence x, and the encoder and the generator 202 may be updated by minimizing a cross-entropy loss as the reconstruction error.

[0045] During the training, m binding peptide sequences may be randomly sampled from the training set 201. A convex combination of the latent codes of the m peptides may be calculated with randomly sampled coefficients, where $2 \leq m \leq K$ and K is a

user-specified hyperparameter. A convex combination may be a positive-weighted linear combination with the sum of the weights equal to 1. The generator 202 generates a binding peptide, and the encoder and generator 202 are updated so that the classifier $q(y|x)$ for the binding class will correctly classify the generated peptide and so the discriminator 204 will classify it as real data.

[0046] Referring now to FIG. 3, a method for developing treatments is shown. Block 302 trains the GAN 200 to generate new binding peptide sequences, using a training dataset that includes both binding and non-binding peptides. From the trained GAN 200, the generator 202 can then generate new binding peptides for a given MHC protein of a pathogen in block 304. Having identified peptides that bind well to the MHC protein of the pathogen, block 306 generates a treatment based on the peptides. Block 308 then treats a patient using the developed treatment, for example by administering a drug that includes the identified peptides, which bind to the MHC protein of the pathogen and encourage the patient's immune system to target the pathogen.

[0047] Referring now to FIG. 4, additional detail on the training of block 302 is shown. Block 402 generates a training dataset. The training dataset may include a set of peptide sequences, each of which may be labeled as binding or non-binding with respect to an MHC protein. Block 403 trains an encoder to convert peptide sequences into a vector embedded in a latent space. As noted above, the encoder maps input peptide sequence x to the space z, including minimizing a kernel maximum mean discrepancy regularization term to enforce a multivariate unit-variance Gaussian distribution. The training of the encoder is performed alongside training the generator 202, as the reconstruction error is used to help minimize a cross-entropy loss.

[0048] Block 404 uses the trained encoder to encode the peptide sequences of the training dataset as vectors. These vectors, in turn, as used as inputs to the generator 202.

Block 408 learns dual projection vectors of the GAN 200. The GAN objective function is optimized with two cross-entropy losses for each classes and with data-specific adaptive weights balancing the discriminator loss and the cross-entropy losses. The generator 202 is updated with tempering softmax outputs to minimize the cross-entropy losses. This training across blocks 403, 404, and 408 is iterated in block 410, with convex combinations of binding sequence embeddings being used to generate binding peptides. The encoder and the generator 202 are updated to fool the discriminator 204 and the classifier. Iteration stops when a maximum number of iterations has been reached.

[0049] Referring now to FIG. 5, an exemplary architecture for the discriminator 204 is shown. A peptide sequence is input as a series of embedded amino acids 502, which are processed by a convolutional layer 504 and one or more fully connected layers 506. The output of the final fully connected layer is a label, indicating whether the input amino acids 502 represent a "real" sequence, present within the training dataset 201, or a sequence that was generated by the generator 202.

[0050] Referring now to FIG. 6, an exemplary architecture for the classifier is shown. As with the discriminator 204, a peptide sequence is input as a series of embedded amino acids 502. The input amino acids 502 are processed by a convolutional layer 604 and one or more fully connected layers 606, trained to identify whether a given peptide sequence binds with an MHC protein. The output of the final fully connected layer is a label, indicating whether the input amino acids 502 represent a binding sequence or a non-binding sequence.

[0051] Referring now to FIG. 7, an exemplary architecture for the generator 700 is shown. Block 702 samples a random noise vector z and a class y as an input to the

generator. This vector may be sampled from a multivariate Gaussian distribution with zero mean and unit diagonal variance, and the binding class label may be fixed.

[0052] The sampled vector and class are processed by one or more fully connected layers 704, which are trained to convert the input into a representation of a peptide sequence. A series of output tempering softmax units 706 processes the output of the fully connected layer(s) 704, generating respective amino acids 502 that, together, form a peptide sequence.

[0053] Referring now to FIG. 8, treatment of a patient 802 is illustrated. A treatment system 804 administers a treatment that is based on a peptide sequence generated by the GAN 200. In particular, a binding peptide may be generated that corresponds to a pathogen or tumor of the patient 102. This binding peptide may be used as part of a treatment that is provided to the patient 102, where the peptide binds to an MHC protein on the pathogen or the tumor cells, helping the patient's autoimmune system identify and remove the pathogen or tumor.

[0054] The administration of the treatment may be overseen by a medical professional 806, who can help connect the treatment system 804. The medical professional 806 may also be involved in the identification of the pathogen or tumor, using diagnostic tools to isolate MHC proteins to be used in identifying binding peptides.

[0055] Referring now to FIG. 9, an exemplary computing device 900 is shown, in accordance with an embodiment of the present invention. The computing device 900 is configured to perform classifier enhancement.

[0056] The computing device 900 may be embodied as any type of computation or computer device capable of performing the functions described herein, including, without limitation, a computer, a server, a rack based server, a blade server, a

workstation, a desktop computer, a laptop computer, a notebook computer, a tablet computer, a mobile computing device, a wearable computing device, a network appliance, a web appliance, a distributed computing system, a processor- based system, and/or a consumer electronic device. Additionally or alternatively, the computing device 900 may be embodied as a one or more compute sleds, memory sleds, or other racks, sleds, computing chassis, or other components of a physically disaggregated computing device.

[0057] As shown in FIG. 9, the computing device 900 illustratively includes the processor 910, an input/output subsystem 920, a memory 930, a data storage device 940, and a communication subsystem 950, and/or other components and devices commonly found in a server or similar computing device. The computing device 900 may include other or additional components, such as those commonly found in a server computer (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 930, or portions thereof, may be incorporated in the processor 910 in some embodiments.

[0058] The processor 910 may be embodied as any type of processor capable of performing the functions described herein. The processor 910 may be embodied as a single processor, multiple processors, a Central Processing Unit(s) (CPU(s)), a Graphics Processing Unit(s) (GPU(s)), a single or multi-core processor(s), a digital signal processor(s), a microcontroller(s), or other processor(s) or processing/controlling circuit(s).

[0059] The memory 930 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 930 may store various data and software used during operation

of the computing device 900, such as operating systems, applications, programs, libraries, and drivers. The memory 930 is communicatively coupled to the processor 910 via the I/O subsystem 920, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor 910, the memory 930, and other components of the computing device 900. For example, the I/O subsystem 920 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, platform controller hubs, integrated control circuitry, firmware devices, communication links (e.g., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.), and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 920 may form a portion of a system-on-a-chip (SOC) and be incorporated, along with the processor 910, the memory 930, and other components of the computing device 900, on a single integrated circuit chip.

[0060] The data storage device 940 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid state drives, or other data storage devices. The data storage device 940 can store program code 940A for model training and program code 940B for generating binding peptides. The communication subsystem 950 of the computing device 900 may be embodied as any network interface controller or other communication circuit, device, or collection thereof, capable of enabling communications between the computing device 900 and other remote devices over a network. The communication subsystem 950 may be configured to use any one or more communication technology (e.g., wired or wireless communications) and associated protocols (e.g., Ethernet, InfiniBand®, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0061] As shown, the computing device 900 may also include one or more peripheral devices 960. The peripheral devices 960 may include any number of additional input/output devices, interface devices, and/or other peripheral devices. For example, in some embodiments, the peripheral devices 960 may include a display, touch screen, graphics circuitry, keyboard, mouse, speaker system, microphone, network interface, and/or other input/output devices, interface devices, and/or peripheral devices.

[0062] Of course, the computing device 900 may also include other elements (not shown), as readily contemplated by one of skill in the art, as well as omit certain elements. For example, various other sensors, input devices, and/or output devices can be included in computing device 900, depending upon the particular implementation of the same, as readily understood by one of ordinary skill in the art. For example, various types of wireless and/or wired input and/or output devices can be used. Moreover, additional processors, controllers, memories, and so forth, in various configurations can also be utilized. These and other variations of the processing system 900 are readily contemplated by one of ordinary skill in the art given the teachings of the present invention provided herein.

[0063] Referring now to FIGs. 10 and 11, exemplary neural network architectures are shown, which may be used to implement parts of the present models. A neural network is a generalized system that improves its functioning and accuracy through exposure to additional empirical data. The neural network becomes trained by exposure to the empirical data. During training, the neural network stores and adjusts a plurality of weights that are applied to the incoming empirical data. By applying the adjusted weights to the data, the data can be identified as belonging to a particular predefined class from a set of classes or a probability that the inputted data belongs to each of the classes can be outputted.

[0064] The empirical data, also known as training data, from a set of examples can be formatted as a string of values and fed into the input of the neural network. Each example may be associated with a known result or output. Each example can be represented as a pair, (x, y), where x represents the input data and y represents the known output. The input data may include a variety of different data types, and may include multiple distinct values. The network can have one input node for each value making up the example's input data, and a separate weight can be applied to each input value. The input data can, for example, be formatted as a vector, an array, or a string depending on the architecture of the neural network being constructed and trained.

[0065] The neural network "learns" by comparing the neural network output generated from the input data to the known values of the examples, and adjusting the stored weights to minimize the differences between the output values and the known values. The adjustments may be made to the stored weights through back propagation, where the effect of the weights on the output values may be determined by calculating the mathematical gradient and adjusting the weights in a manner that shifts the output towards a minimum difference. This optimization, referred to as a gradient descent approach, is a non-limiting example of how training may be performed. A subset of examples with known values that were not used for training can be used to test and validate the accuracy of the neural network.

[0066] During operation, the trained neural network can be used on new data that was not previously used in training or validation through generalization. The adjusted weights of the neural network can be applied to the new data, where the weights estimate a function developed from the training examples. The parameters of the estimated function which are captured by the weights are based on statistical inference.

[0067] In layered neural networks, nodes are arranged in the form of layers. An exemplary simple neural network has an input layer 1020 of source nodes 1022, and a single computation layer 1030 having one or more computation nodes 1032 that also act as output nodes, where there is a single computation node 1032 for each possible category into which the input example could be classified. An input layer 1020 can have a number of source nodes 1022 equal to the number of data values 1012 in the input data 1010. The data values 1012 in the input data 1010 can be represented as a column vector. Each computation node 1032 in the computation layer 1030 generates a linear combination of weighted values from the input data 1010 fed into input nodes 1020, and applies a non-linear activation function that is differentiable to the sum. The exemplary simple neural network can perform classification on linearly separable examples (e.g., patterns).

[0068] A deep neural network, such as a multilayer perceptron, can have an input layer 1020 of source nodes 1022, one or more computation layer(s) 1030 having one or more computation nodes 1032, and an output layer 1040, where there is a single output node 1042 for each possible category into which the input example could be classified. An input layer 1020 can have a number of source nodes 1022 equal to the number of data values 1012 in the input data 1010. The computation nodes 1032 in the computation layer(s) 1030 can also be referred to as hidden layers, because they are between the source nodes 1022 and output node(s) 1042 and are not directly observed. Each node 1032, 1042 in a computation layer generates a linear combination of weighted values from the values output from the nodes in a previous layer, and applies a non-linear activation function that is differentiable over the range of the linear combination. The weights applied to the value from each previous node can be denoted, for example, by w_1, w_2, … w_{n-1}, w_n. The output layer provides the overall response of

the network to the inputted data. A deep neural network can be fully connected, where each node in a computational layer is connected to all other nodes in the previous layer, or may have other configurations of connections between layers. If links between nodes are missing, the network is referred to as partially connected.

[0069] Training a deep neural network can involve two phases, a forward phase where the weights of each node are fixed and the input propagates through the network, and a backwards phase where an error value is propagated backwards through the network and weight values are updated.

[0070] The computation nodes 1032 in the one or more computation (hidden) layer(s) 1030 perform a nonlinear transformation on the input data 1012 that generates a feature space. The classes or categories may be more easily separated in the feature space than in the original data space.

[0071] Embodiments described herein may be entirely hardware, entirely software or including both hardware and software elements. In a preferred embodiment, the present invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

[0072] Embodiments may include a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. A computer-usable or computer readable medium may include any apparatus that stores, communicates, propagates, or transports the program for use by or in connection with the instruction execution system, apparatus, or device. The medium can be magnetic, optical, electronic, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. The medium may include a computer-readable storage medium such as a semiconductor or solid state memory, magnetic tape, a removable

computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk, etc.

[0073] Each computer program may be tangibly stored in a machine-readable storage media or device (e.g., program memory or magnetic disk) readable by a general or special purpose programmable computer, for configuring and controlling operation of a computer when the storage media or device is read by the computer to perform the procedures described herein. The inventive system may also be considered to be embodied in a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner to perform the functions described herein.

[0074] A data processing system suitable for storing and/or executing program code may include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code to reduce the number of times code is retrieved from bulk storage during execution. Input/output or I/O devices (including but not limited to keyboards, displays, pointing devices, etc.) may be coupled to the system either directly or through intervening I/O controllers.

[0075] Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems, cable modem and Ethernet cards are just a few of the currently available types of network adapters.

[0076] As employed herein, the term "hardware processor subsystem" or "hardware processor" can refer to a processor, memory, software or combinations thereof that

cooperate to perform one or more specific tasks. In useful embodiments, the hardware processor subsystem can include one or more data processing elements (e.g., logic circuits, processing circuits, instruction execution devices, etc.). The one or more data processing elements can be included in a central processing unit, a graphics processing unit, and/or a separate processor- or computing element-based controller (e.g., logic gates, etc.). The hardware processor subsystem can include one or more on-board memories (e.g., caches, dedicated memory arrays, read only memory, etc.). In some embodiments, the hardware processor subsystem can include one or more memories that can be on or off board or that can be dedicated for use by the hardware processor subsystem (e.g., ROM, RAM, basic input/output system (BIOS), etc.).

[0077] In some embodiments, the hardware processor subsystem can include and execute one or more software elements. The one or more software elements can include an operating system and/or one or more applications and/or specific code to achieve a specified result.

[0078] In other embodiments, the hardware processor subsystem can include dedicated, specialized circuitry that performs one or more electronic processing functions to achieve a specified result. Such circuitry can include one or more application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), and/or programmable logic arrays (PLAs).

[0079] These and other variations of a hardware processor subsystem are also contemplated in accordance with embodiments of the present invention.

[0080] Reference in the specification to “one embodiment” or “an embodiment” of the present invention, as well as other variations thereof, means that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the appearances of

the phrase "in one embodiment" or "in an embodiment", as well any other variations, appearing in various places throughout the specification are not necessarily all referring to the same embodiment. However, it is to be appreciated that features of one or more embodiments can be combined given the teachings of the present invention provided herein.

[0081] It is to be appreciated that the use of any of the following "/", "and/or", and "at least one of", for example, in the cases of "A/B", "A and/or B" and "at least one of A and B", is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of both options (A and B). As a further example, in the cases of "A, B, and/or C" and "at least one of A, B, and C", such phrasing is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of the third listed option (C) only, or the selection of the first and the second listed options (A and B) only, or the selection of the first and third listed options (A and C) only, or the selection of the second and third listed options (B and C) only, or the selection of all three options (A and B and C). This may be extended for as many items listed.

[0082] The foregoing is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the invention disclosed herein is not to be determined from the Detailed Description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that the embodiments shown and described herein are only illustrative of the present invention and that those skilled in the art may implement various modifications without departing from the scope and spirit of the invention. Those skilled in the art could implement various other feature combinations without departing from the scope and spirit of the invention. Having thus described aspects of the invention, with the details

and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A computer-implemented method of training a model, comprising:

encoding (203) training peptide sequences using an encoder model;

generating (202) a new peptide sequence using a generator model; and

training (206) the encoder model, the generator model, and the discriminator model to cause the generator model to generate new peptides that the discriminator mistakes for the training peptide sequences, including learning projection vectors with respective cross-entropy losses for binding sequences and non-binding sequences.

2. The computer-implemented method of claim 1, wherein the generator model outputs amino acid representations using a plurality of tempering softmax output units.

3. The computer-implemented method of claim 1, wherein generating the new peptide sequence includes sampling a multivariate unit-variate Gaussian distribution as input to the generator.

4. The computer-implemented method of claim 1, wherein the cross-entropy losses include:

$$L_{mi}^{p} = -v_{y}^{p\mathrm{T}}\phi(x^{+}) + \log\sum_{y'} e^{v_{y'}^{p\mathrm{T}}\phi(x^{+})}$$

$$L_{mi}^{q} = -v_{y}^{q\mathrm{T}}\phi(x^{-}) + \log\sum_{y'} e^{v_{y'}^{q\mathrm{T}}\phi(x^{-})}$$

where p corresponds to training peptide sequences, q corresponds to peptide sequences generated by the generator model, v_y^p represents embeddings of the training peptide sequences and generated peptide sequences, v_y^q represents embeddings of the peptide

sequences generated by the generator model, $\phi(\cdot)$ is an embedding function, and $x^{+}\sim P_X$ and $x^{-}\sim Q_X$ are respective training and generated sequences, with P and Q being respective training and generated distributions.

5. The method of claim 1, wherein the encoder model embeds peptide sequences from a training dataset into vectors during training.

6. The method of claim 5, wherein training the encoder model includes minimizing a kernel maximum mean discrepancy regularization term.

7. The method of claim 5, wherein the training dataset includes binding peptide sequences and nonbinding peptide sequences relative to a major histocompatibility complex.

8. The method of claim 5, wherein the generator transforms a binding class label from the encoder and a sampled latent code vector into a peptide feature representation matrix, with each column of the matrix corresponding to an amino acid.

9. The method of claim 1, wherein training the encoder model, the generator model, and the discriminator model uses a loss function that is based on a Wasserstein metric.

10. A computer-implemented method for developing treatments, comprising:

training (302) a generative adversarial network (GAN) model to generate binding peptide sequences relating to a major histocompatibility complex (MHC) protein associated with a virus pathogen or tumor;

generating (304) a new binding peptide sequence using the trained GAN;

developing (306) a treatment for the virus pathogen or tumor associated with the MHC protein using the new binding peptide sequence.

11. The method of claim 10, further comprising treating a person for the virus pathogen or tumor using the developed treatment.

12. A system for training a model, comprising:

a hardware processor (910); and

a memory (940) that stores a computer program, which, when executed by the hardware processor, causes the hardware processor to:

encode (203) training peptide sequences using an encoder model;

generate (202) a new peptide sequence using a generator model; and

train (206) the encoder model, the generator model, and the discriminator model to cause the generator model to generate new peptides that the discriminator mistakes for the training peptide sequences, including learning projection vectors with respective cross-entropy losses for binding sequences and non-binding sequences.

13. The system of claim 12, wherein the generator model outputs amino acid representations using a plurality of tempering softmax output units.

14. The system of claim 12, wherein the computer program further causes the hardware processor to sample a multivariate unit-variate Gaussian distribution as input to the generator.

15. The system of claim 12, wherein the cross-entropy losses include:

$$L_{mi}^{p} = -v_{y}^{p\mathrm{T}}\phi(x^{+}) + \log\sum_{y'} e^{v_{y'}^{p\mathrm{T}}\phi(x^{+})}$$

$$L_{mi}^{q} = -v_{y}^{q\mathrm{T}}\phi(x^{-}) + \log\sum_{y'} e^{v_{y'}^{q\mathrm{T}}\phi(x^{-})}$$

where p corresponds to training peptide sequences, q corresponds to peptide sequences generated by the generator model, v_y^p represents embeddings of the training peptide sequences and generated peptide sequences, v_y^q represents embeddings of the peptide sequences generated by the generator model, $\phi(\cdot)$ is an embedding function, and $x^{+}\sim P_X$ and $x^{-}\sim Q_X$ are respective training and generated sequences, with P and Q being respective training and generated distributions.

16. The system of claim 12, wherein the encoder model embeds peptide sequences from a training dataset into vectors during training.

17. The system of claim 16, wherein the computer program further causes the hardware processor to minimize a kernel maximum mean discrepancy regularization term to train the encoder model.

18. The system of claim 17, wherein the training dataset includes binding peptide sequences and nonbinding peptide sequences relative to a major histocompatibility complex.

19. The system of claim 12, wherein the generator transforms a binding class label from the encoder and a sampled latent code vector into a peptide feature representation matrix, with each column of the matrix corresponding to an amino acid.

20. The system of claim 12, wherein the computer program further causes the hardware processor to use a loss function that is based on a Wasserstein metric to train the encoder model, the generator model, and the discriminator model.

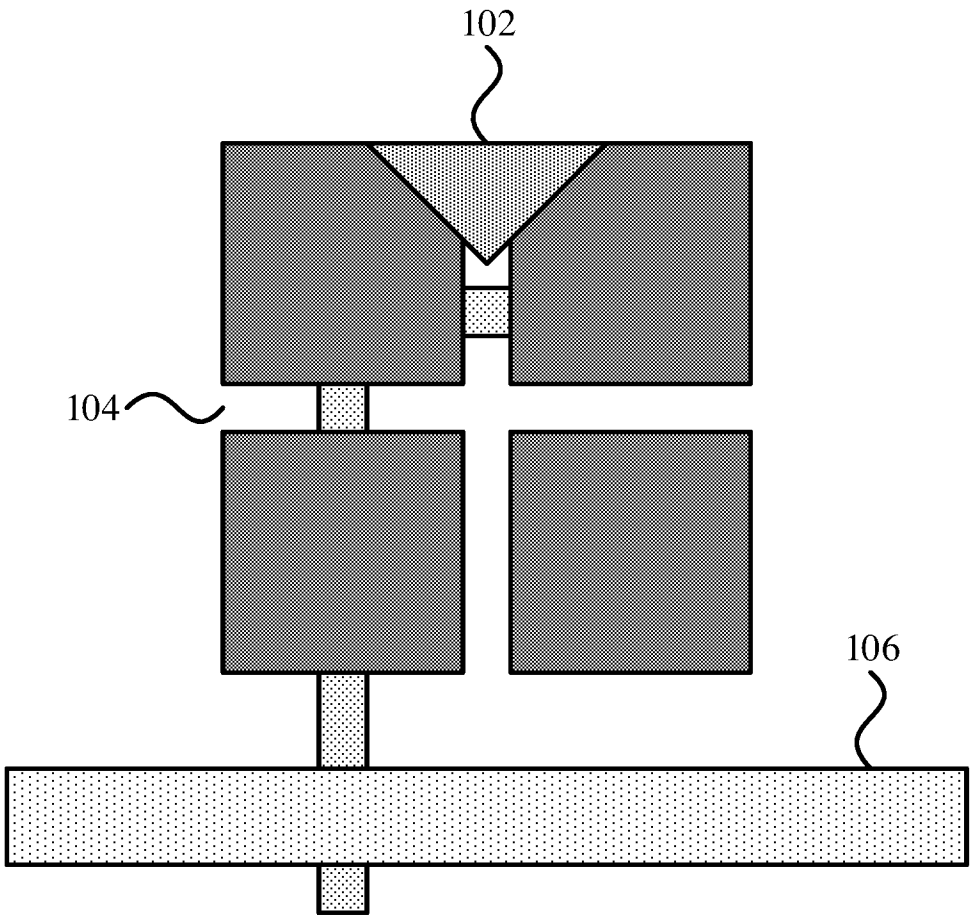


FIG. 1

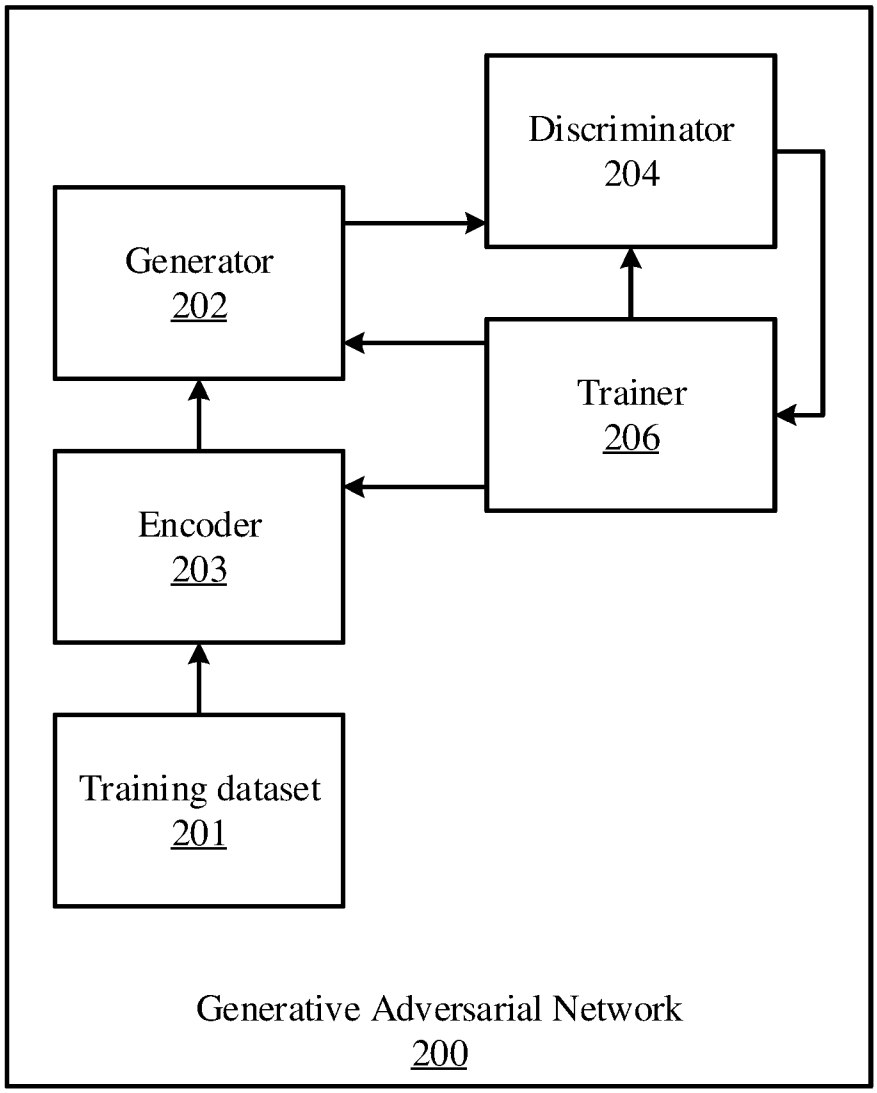

Discriminator
204
Generator
202
Trainer
206
Encoder
203
Training dataset
201
Generative Adversarial Network
200

FIG. 2

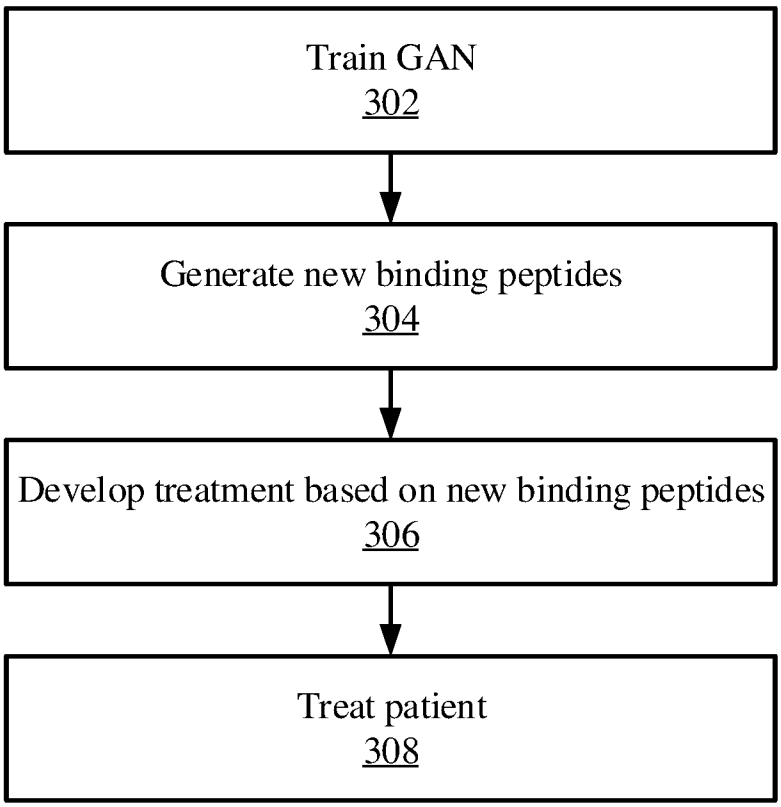

Train GAN
302
Generate new binding peptides
304
Develop treatment based on new binding peptides
306
Treat patient
308

FIG. 3

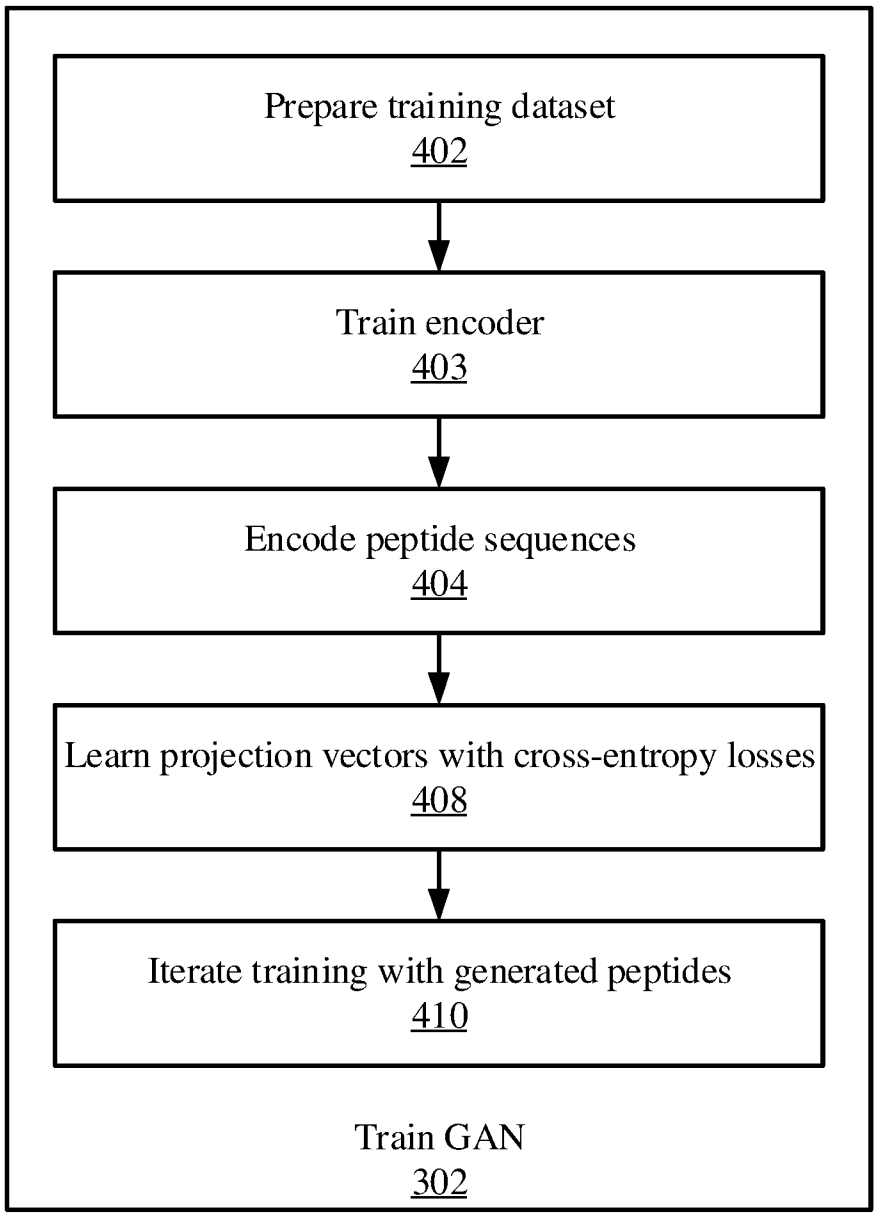

Prepare training dataset
402
Train encoder
403
Encode peptide sequences
404
Learn projection vectors with cross-entropy losses
408
Iterate training with generated peptides
410
Train GAN
302

FIG. 4

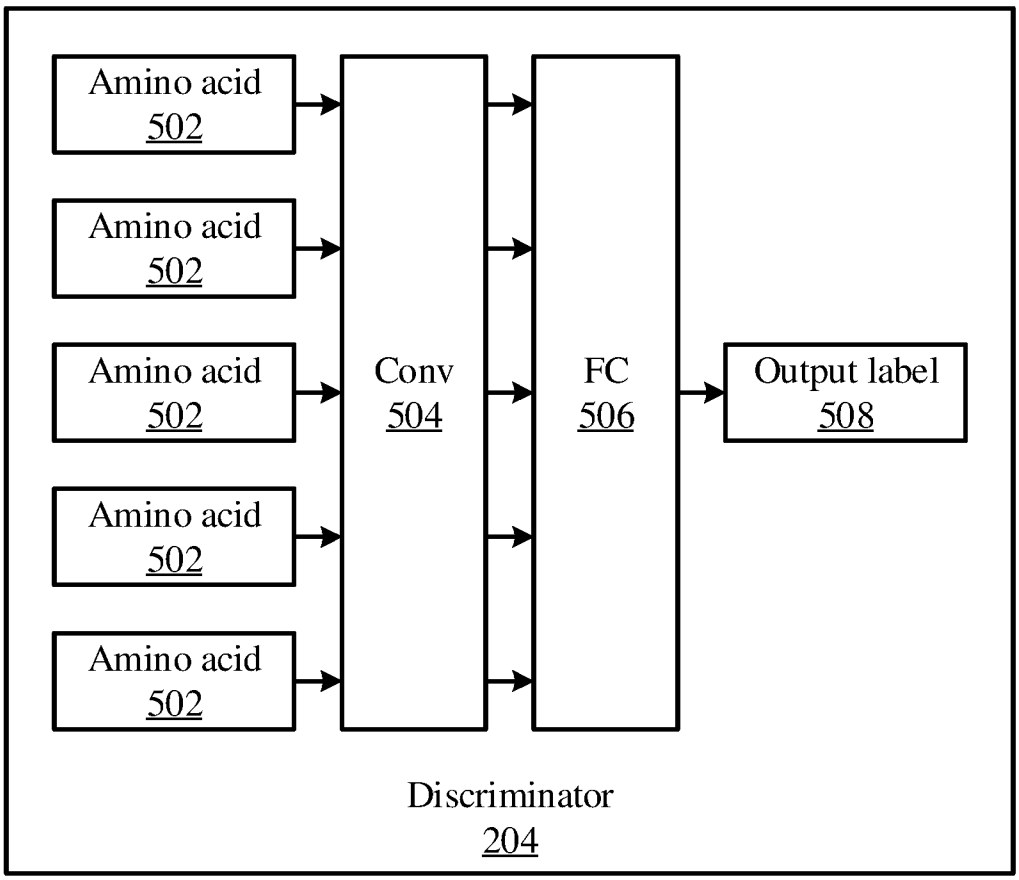

Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Conv
504
FC
506
Output label
508
Discriminator
204

FIG. 5

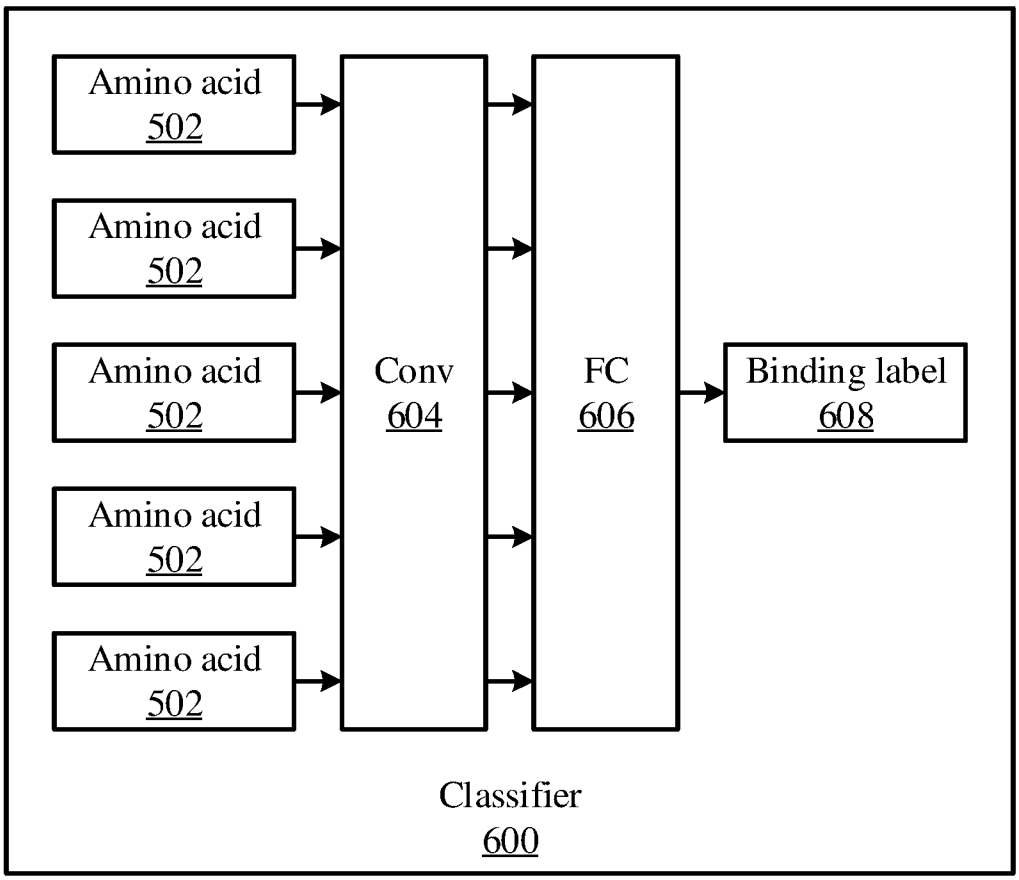

Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Conv
604
FC
606
Binding label
608
Classifier
600

FIG. 6

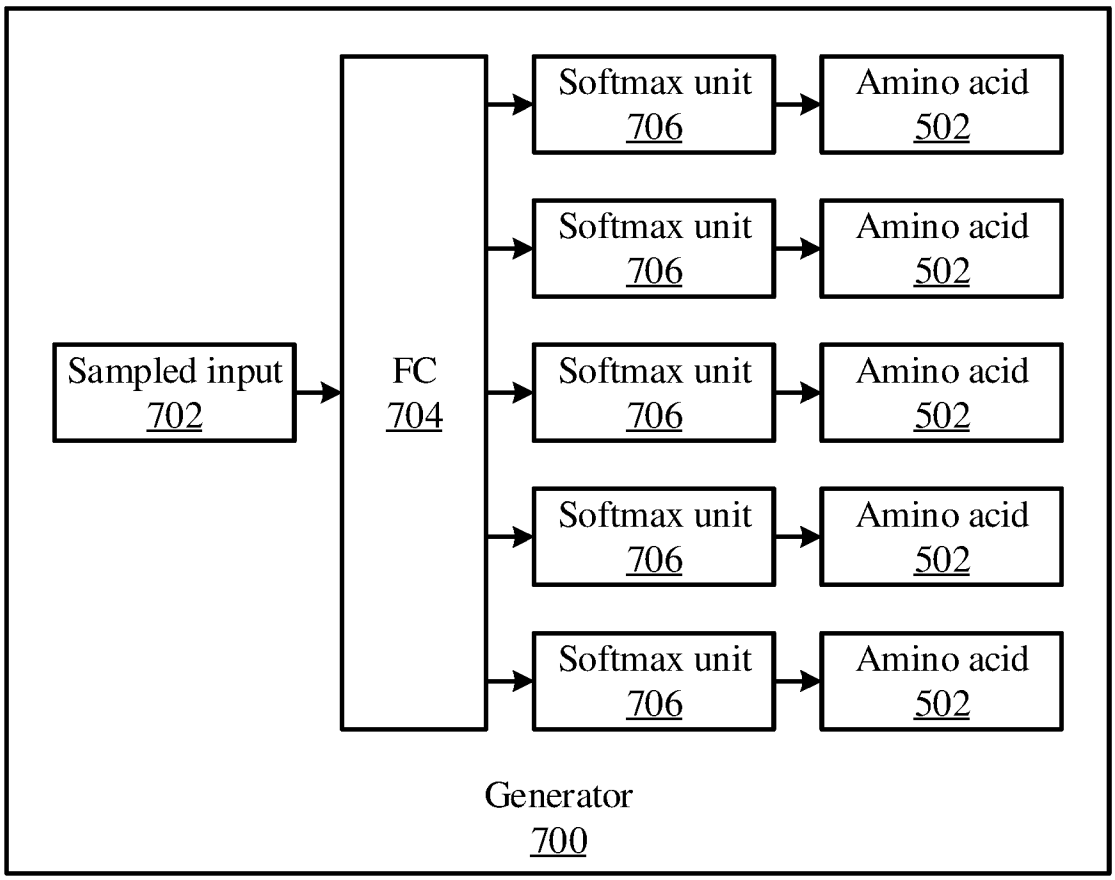

Sampled input
702
FC
704
Softmax unit
706
Softmax unit
706
Softmax unit
706
Softmax unit
706
Softmax unit
706
Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Amino acid
502
Generator
700

FIG. 7

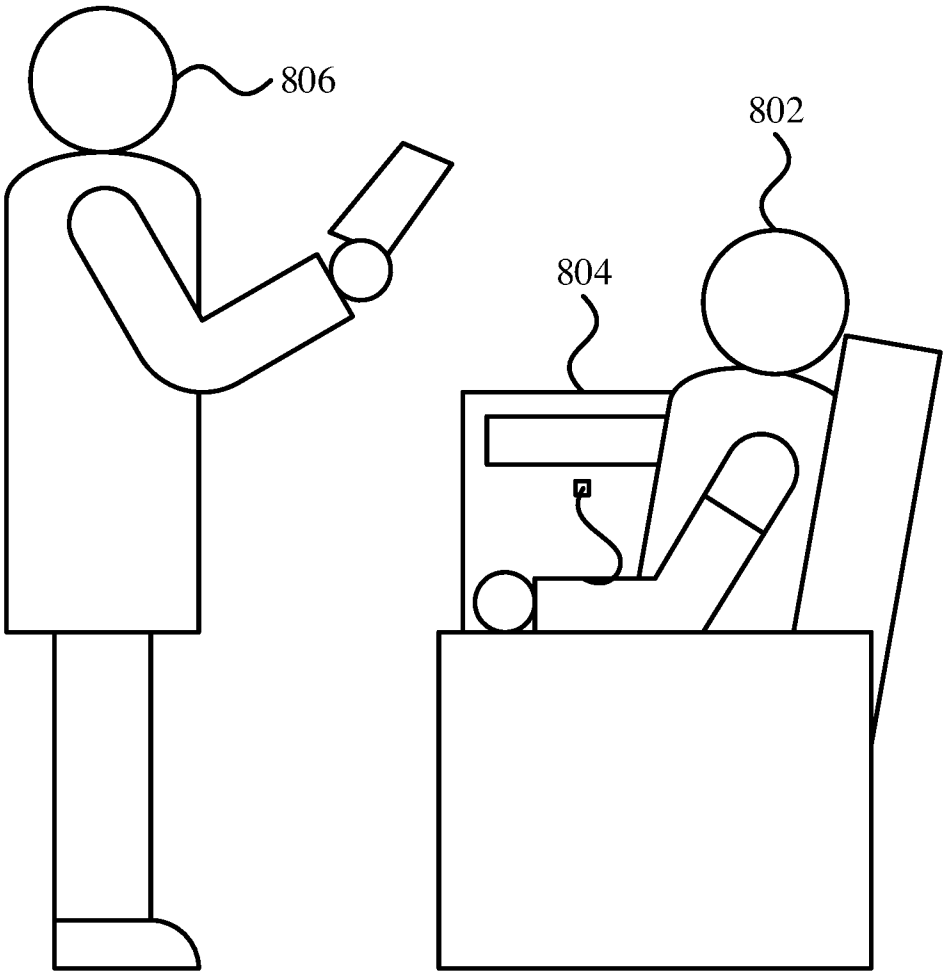


FIG. 8

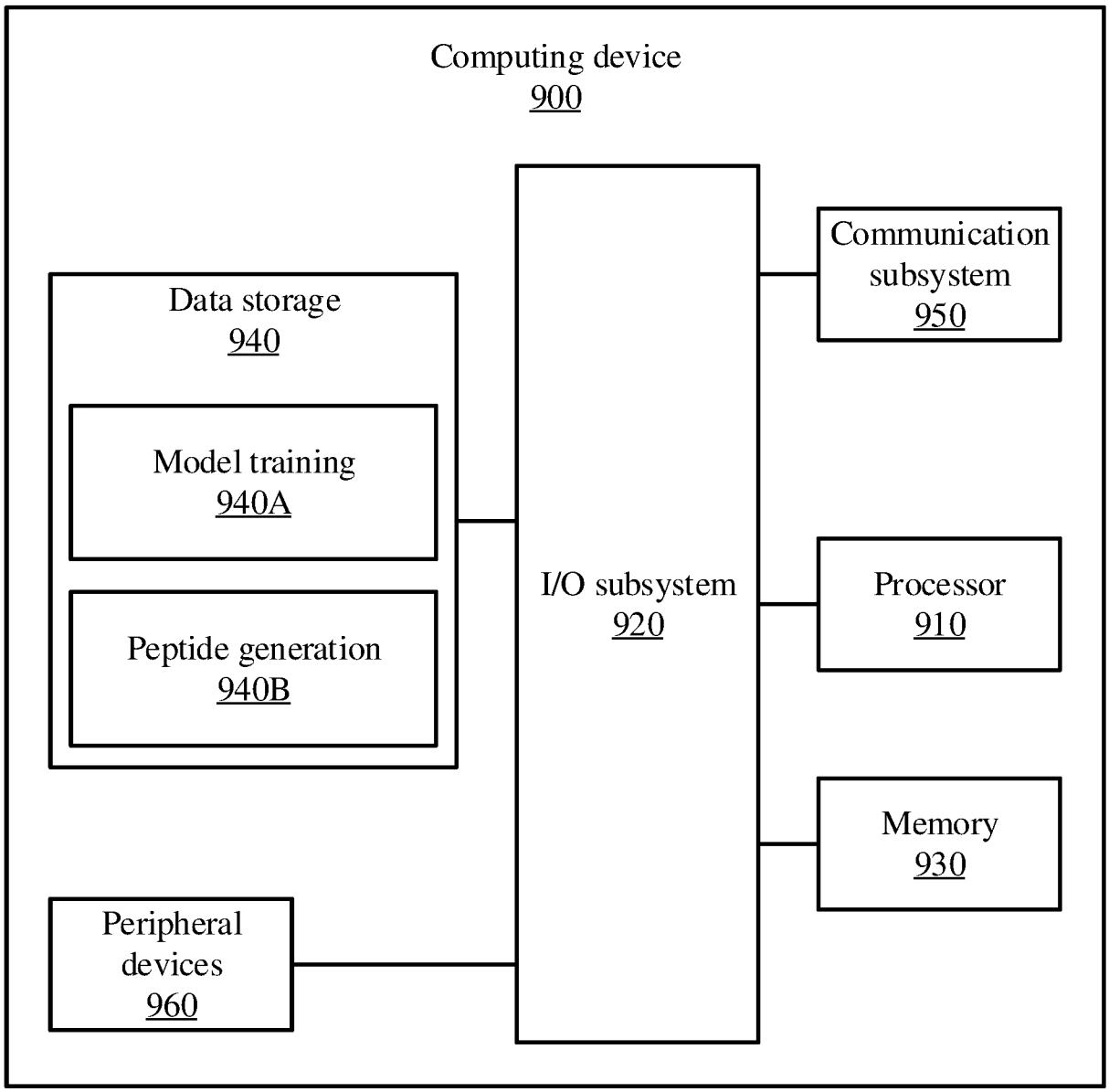

Computing device
900
Data storage
940
Model training
940A
Peptide generation
940B
Peripheral devices
960
I/O subsystem
920
Communication subsystem
950
Processor
910
Memory
930

FIG. 9

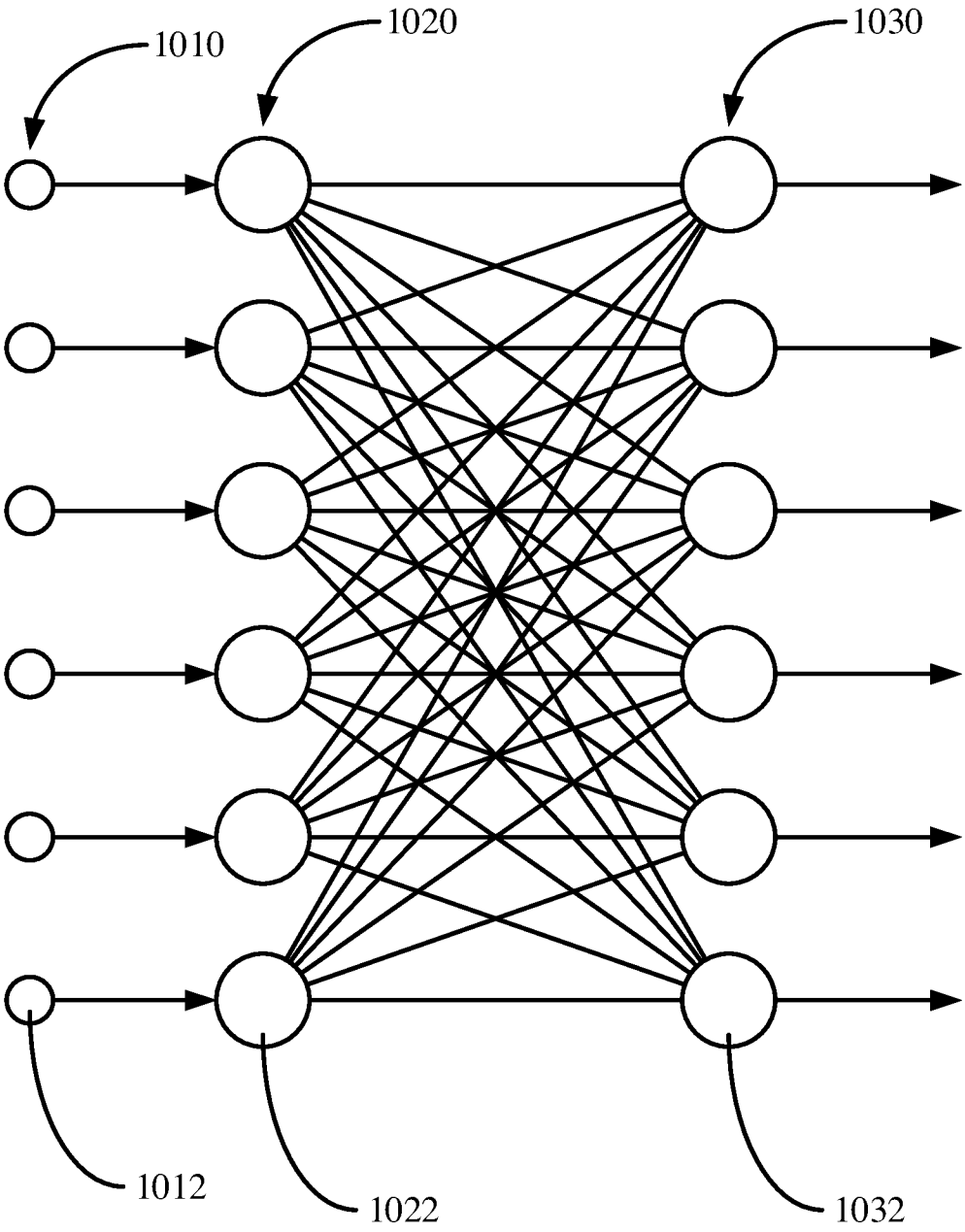


FIG. 10

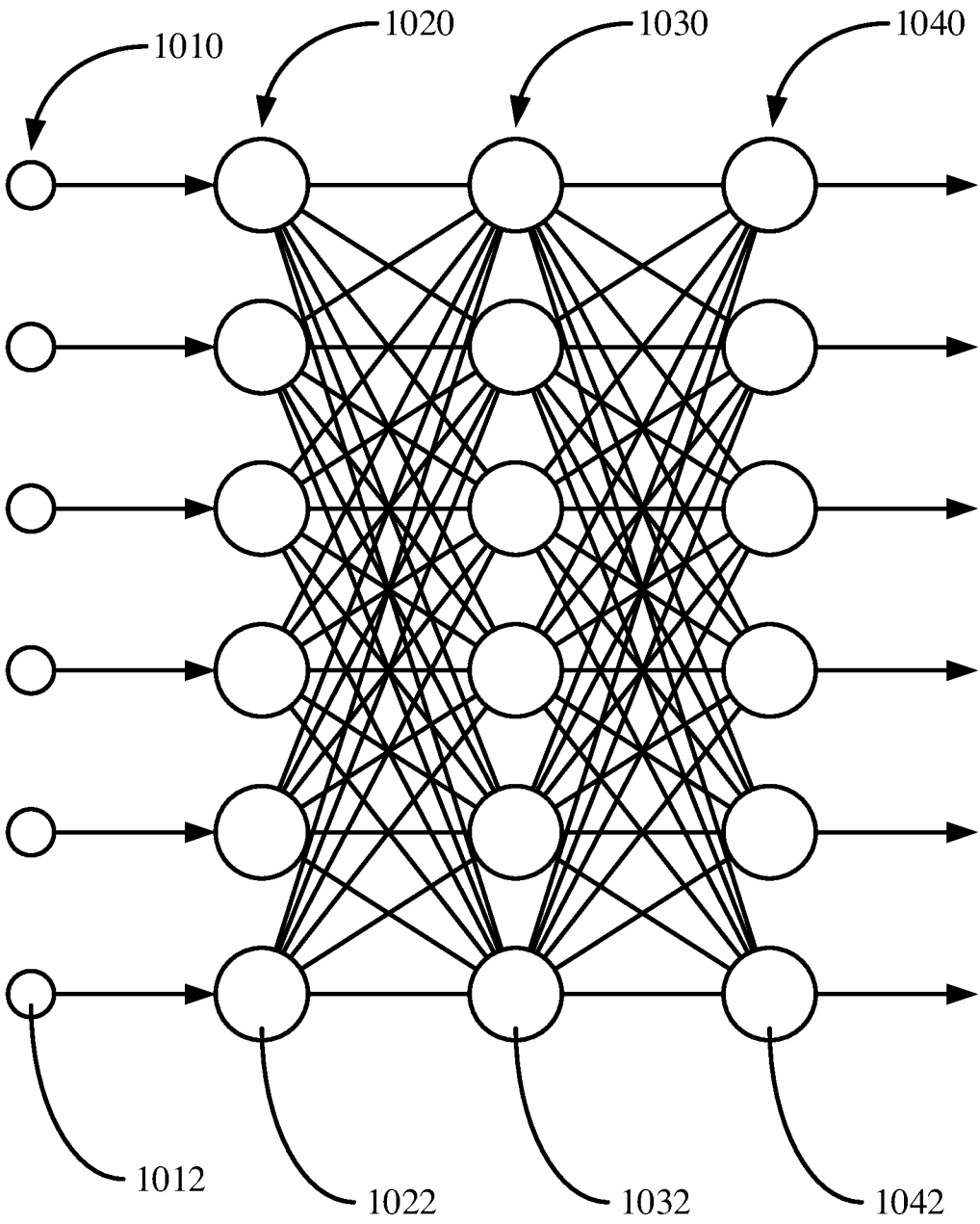


FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2022/023280

A. CLASSIFICATION OF SUBJECT MATTER

G16B 30/00(2019.01)i; **G16B 40/00**(2019.01)i; **G06N 3/08**(2006.01)i; **G06N 3/04**(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G16B 30/00(2019.01); A61B 6/03(2006.01); A61K 39/00(2006.01); G06F 19/00(2011.01); G06N 7/08(2006.01); G06T 7/00(2006.01); G16B 20/00(2019.01); G16B 20/30(2019.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: peptide sequences, encoder model, generator model, discriminator model, cross-entro py losses

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2019-0018933 A1 (PREFERRED NETWORKS, INC.) 17 January 2019 (2019-01-17) claims 1, 17-18, 21	1-20
A	CN 112119464 A (REGENERON PHARMACEUTICALS, INC.) 22 December 2020 (2020-12-22) the entire document	1-20
A	US 2021-0098077 A1 (GRITSTONE ONCOLOGY, INC.) 01 April 2021 (2021-04-01) the entire document	1-20
A	US 2020-0311932 A1 (THE BOARD OF TRUSTEES OF THE LELAND STANFORD JUNIOR UNIVERSITY) 01 October 2020 (2020-10-01) the entire document	1-20
A	WO 2020-236839 A2 (JUST BIOTHERAPEUTICS, INC.) 26 November 2020 (2020-11-26) the entire document	1-20

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 July 2022	**29 July 2022**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**HEO, Joo Hyung**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-5373**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2022/023280

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2019-0018933 A1	17 January 2019	JP 2019-512758 A	16 May 2019
		WO 2017-122785 A1	20 July 2017
CN 112119464 A	22 December 2020	AU 2019-221793 A1	17 September 2020
		AU 2019-221793 A1	22 August 2019
		CA 3091480 A1	22 August 2019
		EP 3753022 A1	23 December 2020
		JP 2021-514086 A	03 June 2021
		JP 7047115 B2	04 April 2022
		KR 10-2020-0125948 A	05 November 2020
		US 2019-0259474 A1	22 August 2019
		WO 2019-161342 A1	22 August 2019
US 2021-0098077 A1	01 April 2021	AU 2016-369519 A1	12 July 2018
		BR 112018012374 A2	04 December 2018
		CA 3008641 A1	22 June 2017
		CN 108601731 A	28 September 2018
		CO 2018007417 A2	28 September 2018
		EP 3389630 A1	24 October 2018
		JP 2019-501967 A	24 January 2019
		PE 15352018 A1	26 September 2018
		PE 20181535 A1	26 September 2018
		PH 12018501267 A1	11 February 2019
		RU 2018124997 A	16 January 2020
		RU 2018124997 A3	29 May 2020
		RU 2729116 C2	04 August 2020
		SG 11201804957 A	30 July 2018
		TW 201733598 A	01 October 2017
		US 10055540 B2	21 August 2018
		US 10847252 B2	24 November 2020
		US 10847253 B2	24 November 2020
		US 1183286 B2	23 November 2021
		US 2017-0199961 A1	13 July 2017
		US 2017-0212984 A1	27 July 2017
		US 2018-0330055 A1	15 November 2018
		US 2019-0034585 A1	31 January 2019
		US 2019-0065675 A1	28 February 2019
		US 2021-0166784 A1	03 June 2021
		WO 2017-106638 A1	22 June 2017
US 2020-0311932 A1	01 October 2020	None	
WO 2020-236839 A2	26 November 2020	EP 3956896 A2	23 February 2022
		KR 10-2022-0011148 A	27 January 2022
		WO 2020-236839 A3	10 June 2021