

**EFFICIENT ARTIFICIAL INTELLIGENCE-BASED BASE
CALLING OF INDEX SEQUENCES**

PRIORITY APPLICATIONS

[0001] This application claims priority of U.S. Nonprovisional Patent Application No. 17/839,331, titled, “EFFICIENT ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed June 13, 2022 (Attorney Docket No. ILLM 1046-2/IP-2135-US), which claims the benefit of U.S. Provisional Patent Application No. 63/217,644, titled, “EFFICIENT ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed July 1, 2021 (Attorney Docket No. ILLM 1046-1/IP-2135-PRV). The priority applications are hereby incorporated by reference for all purposes.

FIELD OF THE TECHNOLOGY DISCLOSED

[0002] The technology disclosed relates to artificial intelligence type computers and digital data processing systems and corresponding data processing methods and products for emulation of intelligence (*i.e.*, knowledge-based systems, reasoning systems, and knowledge acquisition systems); and including systems for reasoning with uncertainty (*e.g.*, fuzzy logic systems), adaptive systems, machine learning systems, and artificial neural networks. In particular, the technology disclosed relates to using deep neural networks such as deep convolutional neural networks for analyzing data.

INCORPORATIONS BY REFERENCE

[0003] The following are hereby incorporated by reference as if fully set forth herein:

[0004] U.S. Patent Application No. 62/979,384, titled “ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed 20 February 2020 (Attorney Docket No. ILLM 1015-1/IP-1857-PRV);

[0005] U.S. Patent Application No. 62/979,414, titled “ARTIFICIAL INTELLIGENCE-BASED MANY-TO-MANY BASE CALLING,” filed 20 February 2020 (Attorney Docket No. ILLM 1016-1/IP-1858-PRV);

[0006] U.S. Patent Application No. 62/979,385, titled “KNOWLEDGE DISTILLATION-BASED COMPRESSION OF ARTIFICIAL INTELLIGENCE-BASED BASE CALLER,” filed 20 February 2020 (Attorney Docket No. ILLM 1017-1/IP-1859-PRV);

[0007] U.S. Patent Application No. 63/072,032, titled “DETECTING AND FILTERING CLUSTERS BASED ON ARTIFICIAL INTELLIGENCE-PREDICTED BASE CALLS,” filed 28 August 2020 (Attorney Docket No. ILLM 1018-1/IP-1860-PRV);

[0008] U.S. Patent Application No. 62/979,412, titled “MULTI-CYCLE CLUSTER BASED REAL TIME ANALYSIS SYSTEM,” filed 20 February 2020 (Attorney Docket No. ILLM 1020-1/IP-1866-PRV);

[0009] U.S. Patent Application No. 16/825,987, titled “TRAINING DATA GENERATION FOR ARTIFICIAL INTELLIGENCE-BASED SEQUENCING,” filed 20 March 2020 (Attorney Docket No. ILLM 1008-16/IP-1693-US);

[0010] U.S. Patent Application No. 16/825,991 titled “ARTIFICIAL INTELLIGENCE-BASED GENERATION OF SEQUENCING METADATA,” filed 20 March 2020 (Attorney Docket No. ILLM 1008-17/IP-1741-US);

[0011] U.S. Patent Application No. 16/826,126, titled “ARTIFICIAL INTELLIGENCE-BASED BASE CALLING,” filed 20 March 2020 (Attorney Docket No. ILLM 1008-18/IP-1744-US);

[0012] U.S. Patent Application No. 16/826,134, titled “ARTIFICIAL INTELLIGENCE-BASED QUALITY SCORING,” filed 20 March 2020 (Attorney Docket No. ILLM 1008-19/IP-1747-US);

[0013] U.S. Patent Application No. 16/826,168, titled “ARTIFICIAL INTELLIGENCE-BASED SEQUENCING,” filed 21 March 2020 (Attorney Docket No. ILLM 1008-20/IP-1752-PRV-US); and

[0014] U.S. Patent Application No. 17/179,395, titled “DATA COMPRESSION FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLING,” filed 18 February 2021 (Attorney Docket No. ILLM 1029-2/IP-1964-US).

BACKGROUND

[0015] The subject matter discussed in this section cannot and should not be assumed to be prior art merely as a result of its mention in this section because, among other things, it may represent different approaches, which in and of themselves, may also correspond to implementations of the claimed technology. Similarly, a problem mentioned, or identified, in this section or associated with the subject matter provided as background, cannot and should not be assumed to have been previously recognized in the prior art as, among other things, it may also be part of the solution proposed.

[0016] Improvements in next-generation sequencing (NGS) technology have greatly increased sequencing speed and data output, resulting in relatively massive sample throughput of current sequencing platforms. Approximately ten years ago, the Illumina Genome Analyzer™ was capable of generating up to one gigabyte of sequence data per run. Today, the Illumina

NovaSeq™ series are capable of generating up to two terabytes of data in two days, representing a greater than 2000x increase in capacity.

[0017] One aspect of realizing this increased capacity is multiplexing, which adds unique sequences, called indexes, to each DNA fragment during library preparation. This allows large numbers of libraries to be pooled and sequenced simultaneously during a single sequencing run. Gains in throughput from multiplexing come with an added layer of complexity, as sequencing reads from pooled libraries need to be identified and sorted computationally in a process called demultiplexing before final data analysis. Index misassignment between multiplexed libraries is a known issue that has impacted NGS technologies from the time sample multiplexing was developed.

[0018] An opportunity arises to use artificial intelligence and neural networks to base call index sequences. Higher base calling throughput and increased base calling accuracy may result.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] In the drawings, like reference characters generally refer to like parts throughout the different views. Also, the drawings are not necessarily to scale, with an emphasis instead generally being placed upon illustrating the principles of the technology disclosed. In the following description, various implementations of the technology disclosed are described with reference to the following drawings, in which:

[0020] Figure 1 shows one implementation of the sequencing of polynucleotides from indexed libraries.

[0021] Figure 2 shows one implementation of sequencing a target sequence to generate a target read and sequencing an index sequence to generate an index read.

[0022] Figure 3 illustrates one implementation of the disclosed neural network-based base caller.

[0023] Figures 4A, 4B and 4C depict one implementation of shuffling the index images for processing by the disclosed neural network-based base caller.

[0024] Figures 5A, 5B and 5C depict another implementation of shuffling the index images for processing by the disclosed neural network-based base caller.

[0025] Figure 6 depicts a subset selector configured to effectively generate a reordering of a set of sequencing images (reordered set of sequencing images) as a reordered subset of sequencing images, based on two parameters in accordance with one implementation.

[0026] Figures 7A, 7B and 7C depict a reordered subset of sequencing images (also shown in Figure 6) in connection with the neural network-based base caller (also shown in Figure 4).

[0027] Figures 8A and 8B depict additional examples of reordered subsets of sequencing images in connection with a neural network-based base caller in accordance with additional implementations.

[0028] Figure 9 depicts initial (first) training and further (second) training in accordance with two implementations.

[0029] Figure 10 depicts a method for training a neural network-based base caller in accordance with one implementation.

[0030] Figure 11 depicts another method for training a neural network-based base caller in accordance with another implementation.

[0031] Figure 12 depicts a method for training a neural network-based base caller for base calling of sequencing images, in accordance with one implementation.

[0032] Figure 13 depicts another method for training a neural network-based base caller in accordance with one implementation.

[0033] Figure 14 shows a flow chart of a system that executes the disclosed base calling, in accordance with one implementation.

[0034] Figure 15 shows measured base calling error rates for the previously disclosed “augmentation” and “normalization” implementations of the neural network-based base caller 300, as applied on 1-Plex sequencing data.

[0035] Figures 16A, B and C compare measured base calling error rates for three different implementations of the neural network-based base caller 300, as applied on different complexities (*e.g.*, 96-Plex, 1-Plex, and 3-Plex).

[0036] Figures 17A, B and C compare measured base calling error rates for three different implementations of the neural network-based base caller 300, as applied on different complexities (*e.g.*, 4-Plex, 2-Plex, and 1-Plex).

[0037] Figure 18 depicts a plot that compares the “demultiplex” results for tiles from runs with different Plexities for RTA (blue), a previously disclosed deep learning implementation (orange) (DL 3c Aug), and the currently disclosed joint training and shuffling implementation (green) (DL 5c Norm*).

[0038] Figures 19A and 19B depict one implementation of a sequencing system.

[0039] Figure 19C is a simplified block diagram of a system for the analysis of sensor data from the sequencing system, such as base call sensor outputs.

[0040] Figure 20A is a simplified diagram showing aspects of the base calling operation, including functions of a runtime program executed by a host processor.

[0041] Figure 20B is a simplified diagram of a configuration of a configurable processor such as that of Figure 19C.

[0042] Figure 21A illustrates one implementation of jointly training the neural network-based base caller 300 on reordered index images and sequencing images.

[0043] Figure 21B illustrates one implementation of how the reordered index images 2132 of Figure 21A are generated.

[0044] Figure 22 is a computer system that can be used by the sequencing system of Figure 19A to implement the base calling techniques disclosed herein.

[0045] Figure 23 depicts an example of 96 indexes.

[0046] Figures 24 and 25 show various implementations of the technology disclosed.

DETAILED DESCRIPTION

[0047] The following discussion is presented to enable any person skilled in the art to make and use the technology disclosed and is provided in the context of a particular application and its requirements. Various modifications to the disclosed implementations will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other implementations and applications without departing from the spirit and scope of the technology disclosed. Thus, the technology disclosed is not intended to be limited to the implementations shown but is to be accorded the widest scope consistent with the principles and features disclosed herein.

Multiplexing

[0048] Figure 1 shows one implementation of the sequencing of polynucleotides from indexed libraries. When polynucleotides from different libraries are pooled or multiplexed for sequencing, the polynucleotides from each library are modified to include a library-specific index sequence. During sequencing, the index sequences are sequenced along with target polynucleotide sequences from the libraries. An index sequence is associated with a target polynucleotide sequence so that the library from which the target sequence originated can be identified.

[0049] Panel A shows indexed libraries 102. Here, unique index sequences (“indexes”) are added to two different libraries during library preparation. The first index sequence (Index A) has a barcode of “CATTCG.” The second index sequence (Index B) has a barcode of “AACTGA.”

[0050] Panel B shows pooling 104. Here, the indexed libraries 102 are pooled together and loaded into the same flow cell lane.

[0051] Panel C shows sequencing 106 and the sequencing output 116. Here, the indexed libraries 102 are sequenced together during a single instrument run. All sequences are then

exported to an output file 116. The output file 116 comprises sequence reads (in green) coupled to corresponding index reads (in blue and magenta).

[0052] Panel D shows demultiplexing 108. Here, a demultiplexing algorithm sorts the sequence reads into different files according to their indexes.

[0053] Panel E shows alignment 110. Here, each set of the demultiplexed sequence reads is aligned to the appropriate reference sequence.

[0054] Additional details about multiplexing, index sequences, and demultiplexing can be found in Illumina, “Indexed Sequencing Overview Guide,” Document No. 15057455, v. 5, March 2019 and in Illumina’s patent application publications US 2018/0305751, US 2018/0334712, US 2016/0110498, US 2018/0334711, and WO 2019/090251, each of which is incorporated herein by reference.

Target Sequences and Index Sequences

[0055] Figure 2 shows one implementation of sequencing a target sequence 222 to generate a target read 202 (“GTCCGATA”) and sequencing an index sequence 232 to generate an index read 204 (“AACTGA”). The index sequence 232 can be a synthetic sequence of nucleotides that is coupled to the target sequence 222 during the template preparation step. The target sequence 222 can be naturally occurring DNA, RNA, or some other biological molecule. The length of the index sequence 232 can range from two to twenty nucleotides. For example, the index sequence 232 can be one to ten nucleotides long or four to six nucleotides long. A four-nucleotide index sequence gives the possibility of multiplexing 256 samples on the same array. A six-nucleotide index sequence enables 4096 samples to be processed on the same array.

[0056] During the sequencing 106, a target primer 212 traverses the target sequence 222 and produces the target read 202 (“GTCCGATA”) and an index primer 224 traverses the index sequence 232 and produces the index read 204 (“AACTGA”). In some implementations, the sequencing 106 is Illumina’s single-indexed sequencing. In other implementations, the sequencing 106 is Illumina’s dual-indexed sequencing.

[0057] Base calling is the process of determining the nucleotide composition of the target sequence 222 and the index sequence 232, *i.e.*, the process of generating the target read 202 (“GTCCGATA”) and the index read 204 (“AACTGA”).

Sequencing Images

[0058] Base calling involves analyzing image data, *i.e.*, sequencing images, produced during a sequencing run (or sequencing reaction) carried out by a sequencing instrument such as

Illumina's iSeq, HiSeqX, HiSeq 3000, HiSeq 4000, HiSeq 2500, NovaSeq 6000, NextSeq 550, NextSeq 1000, NextSeq 2000, NextSeqDx, MiSeq, and MiSeqDx.

[0059] The following discussion outlines how the sequencing images are generated and what they depict, in accordance with one implementation.

[0060] Base calling decodes the intensity data encoded in the sequencing images into nucleotide sequences. In one implementation, the Illumina sequencing platforms employ cyclic reversible termination (CRT) chemistry for base calling. The process relies on growing nascent strands complementary to template strands with fluorescently-labeled nucleotides, while tracking the emitted signal of each newly added nucleotide. The fluorescently-labeled nucleotides each have a 3' removable block that anchors a fluorophore signal of the nucleotide type.

[0061] Sequencing occurs in repetitive cycles, each comprising three steps: (a) extension of a nascent strand by adding the fluorescently-labeled nucleotide; (b) excitation of the fluorophore using one or more lasers of an optical system of the sequencing instrument and imaging through different filters of the optical system, yielding the sequencing images; and (c) cleavage of the fluorophore and removal of the 3' block in preparation for the next sequencing cycle.

Incorporation and imaging cycles are repeated up to a designated number of sequencing cycles, defining the read length. Using this approach, each cycle interrogates a new position along the template strands.

[0062] The tremendous power of the Illumina sequencers stems from their ability to simultaneously execute and sense millions or even billions of clusters (also called "analytes") undergoing CRT reactions. A cluster comprises approximately one thousand identical copies of a template strand, though clusters vary in size and shape. The clusters are grown from the template strand, prior to the sequencing run, by bridge amplification or exclusion amplification of the input library. The purpose of the amplification and cluster growth is to increase the intensity of the emitted signal since the imaging device cannot reliably sense the fluorophore signal of a single strand. However, the physical distance of the strands within a cluster is small, so the imaging device perceives the cluster of strands as a single spot.

[0063] Sequencing occurs in a flow cell (or biosensor) – a small glass slide that holds the input strands. The flow cell is connected to the optical system, which comprises microscopic imaging, excitation lasers, and fluorescence filters. The flow cell comprises multiple chambers called lanes. The lanes are physically separated from each other and may contain different tagged sequencing libraries, distinguishable without sample cross contamination. In some implementations, the flow cell comprises a patterned surface. A "patterned surface" refers to an arrangement of different regions in or on an exposed layer of a solid support.

[0064] The imaging device of the sequencing instrument (*e.g.*, a solid-state imager such as a charge-coupled device (CCD) or a complementary metal–oxide–semiconductor (CMOS) sensor) takes snapshots at multiple locations along the lanes in a series of non-overlapping regions called tiles. For example, there can be sixty-four or ninety-six tiles per lane. A tile holds hundreds of thousands to millions of clusters.

[0065] The output of the sequencing run is the sequencing images. Sequencing images depict intensity emissions of the clusters and their surrounding background using a grid (or array) of pixelated units (*e.g.*, pixels, superpixels, subpixels). The intensity emissions are stored as intensity values of the pixelated units. The sequencing images have dimensions $w \times h$ of the grid of pixelated units, where w (width) and h (height) are any numbers ranging from 1 to 100,000 (*e.g.*, 115×115 , 200×200 , 1800×2000 , 2200×25000 , 2800×3600 , 4000×400). In some implementations, w and h are the same. In other implementations, w and h are different.

[0066] Those sequencing cycles of the sequencing run that sequence the target sequence 222 are called “target sequencing cycles” and those sequencing cycles of the sequencing run that sequence the index sequence 232 are called “index sequencing cycles.” The sequencing images generated during the target sequencing cycles are called “target images” and the sequencing images generated during the index sequencing cycles are called “index images.”

Neural Network-Based Base Calling

[0067] The following discussion focuses on a neural network-based base caller 300 described herein. First, the input to the neural network-based base caller 300 is described, in accordance with one implementation. Then, examples of the structure and form of the neural network-based base caller 300 are provided. Finally, the output of the neural network-based base caller 300 is described, in accordance with one implementation.

[0068] A data flow logic provides the sequencing images to the neural network-based base caller 300 for base calling. The neural network-based base caller 300 accesses the sequencing images on a patch-by-patch basis (or a tile-by-tile basis). Each of the patches is a sub-grid (or sub-array) of pixelated units in the grid of pixelated units that forms the sequencing images. The patches have dimensions $q \times r$ of the sub-grid of pixelated units, where q (width) and r (height) are any numbers ranging from 1 to 10000 (*e.g.*, 3×3 , 5×5 , 7×7 , 10×10 , 15×15 , 25×25 , 64×64 , 78×78 , 115×115). In some implementations, q and r are the same. In other implementations, q and r are different. In some implementations, the patches extracted from a sequencing image are of the same size. In other implementations, the patches are of different sizes. In some implementations, the patches can have overlapping pixelated units (*e.g.*, on the edges).

[0069] Sequencing produces m sequencing images per sequencing cycle for a corresponding m image channels. That is, each of the sequencing images has one or more image (or intensity) channels (analogous to the red, green, blue (RGB) channels of a color image). In one implementation, each image channel corresponds to one of a plurality of filter wavelength bands. In another implementation, each image channel corresponds to one of a plurality of imaging events in a sequencing cycle. In yet another implementation, each image channel corresponds to a combination of illumination with a specific laser and imaging through a specific optical filter. The image patches are tiled (or accessed) from each of the m image channels for a particular sequencing cycle. In different implementations such as 4-, 2-, and 1-channel chemistries, m is 4 or 2. In other implementations, m is 1, 3, or greater than 4.

[0070] Consider, for example, that a sequencing run is implemented using two different image channels: a blue channel and a green channel. Then, at each sequencing cycle, the sequencing run produces a blue image and a green image. This way, for a series of k sequencing cycles of the sequencing run, a sequence of k pairs of blue and green images is produced as output and stored as the sequencing images. Accordingly, a sequence of k pairs of blue and green image patches is generated for the patch-level processing by the neural network-based base caller 300.

[0071] The input image data to the neural network-based base caller 300 for a single iteration of base calling (or a single instance of forward pass or a single forward traversal) comprises data for a sliding window of multiple sequencing cycles. The sliding window can include, for example, a current sequencing cycle, one or more preceding sequencing cycles, and one or more successive sequencing cycles.

[0072] In one implementation, the input image data comprises data for three sequencing cycles, such that data for a current (time t) sequencing cycle to be base called is accompanied with (i) data for a left-flanking/context/previous/preceding/prior (time $t-1$) sequencing cycle and (ii) data for a right-flanking/context/next/successive/subsequent (time $t+1$) sequencing cycle.

[0073] In another implementation, the input image data comprises data for five sequencing cycles, such that data for a current (time t) sequencing cycle to be base called is accompanied with (i) data for a first left-flanking/context/previous/preceding/prior (time $t-1$) sequencing cycle, (ii) data for a second left-flanking/context/previous/preceding/prior (time $t-2$) sequencing cycle, (iii) data for a first right-flanking/context/next/successive/subsequent (time $t+1$) sequencing cycle, and (iv) data for a second right-flanking/context/next/successive/subsequent (time $t+2$) sequencing cycle.

[0074] In yet another implementation, the input image data comprises data for seven sequencing cycles, such that data for a current (time t) sequencing cycle to be base called is

accompanied with (i) data for a first left-flanking/context/previous/preceding/prior (time $t-1$) sequencing cycle, (ii) data for a second left-flanking/context/previous/preceding/prior (time $t-2$) sequencing cycle, (iii) data for a third left-flanking/context/previous/preceding/prior (time $t-3$) sequencing cycle, (iv) data for a first right-flanking/context/next/successive/subsequent (time $t+1$) sequencing cycle, (v) data for a second right-flanking/context/next/successive/subsequent (time $t+2$) sequencing cycle, and (vi) data for a third right-flanking/context/next/successive/subsequent (time $t+3$) sequencing cycle. In other implementations, the input image data comprises data for a single sequencing cycle. In yet other implementations, the input image data comprises data for 10, 15, 20, 30, 58, 75, 92, 130, 168, 175, 209, 225, 230, 275, 318, 325, 330, 525, or 625 sequencing cycles.

[0075] The neural network-based base caller 300 processes the image patches through its convolution layers and produces an alternative representation, according to one implementation. The alternative representation is then used by an output layer (*e.g.*, a softmax layer) to generate a base call for either just the current (time t) sequencing cycle or each of the sequencing cycles, *i.e.*, the current (time t) sequencing cycle, the first and second preceding (time $t-1$, time $t-2$) sequencing cycles, and the first and second succeeding (time $t+1$, time $t+2$) sequencing cycles. The resulting base calls form the sequencing reads.

[0076] In one implementation, the neural network-based base caller 300 outputs a base call for a single target cluster for a particular sequencing cycle. In another implementation, the neural network-based base caller 300 outputs a base call for each target cluster in a plurality of target clusters for the particular sequencing cycle. In yet another implementation, the neural network-based base caller 300 outputs a base call for each target cluster in a plurality of target clusters for each sequencing cycle in a plurality of sequencing cycles, thereby producing a base call sequence for each target cluster.

[0077] In one implementation, the neural network-based base caller 300 is a multilayer perceptron (MLP). In another implementation, the neural network-based base caller 300 is a feedforward neural network. In yet another implementation, the neural network-based base caller 300 is a fully-connected neural network. In a further implementation, the neural network-based base caller 300 is a fully convolutional neural network. In yet further implementation, the neural network-based base caller 300 is a semantic segmentation neural network. In yet another further implementation, the neural network-based base caller 300 is a generative adversarial network (GAN).

[0078] In one implementation, the neural network-based base caller 300 is a convolutional neural network (CNN) with a plurality of convolution layers. In another implementation, the neural network-based base caller 300 is a recurrent neural network (RNN) such as a long short-

term memory network (LSTM), bi-directional LSTM (Bi-LSTM), or a gated recurrent unit (GRU). In yet another implementation, the neural network-based base caller 300 includes both a CNN and an RNN.

[0079] In yet other implementations, the neural network-based base caller 300 can use 1D convolutions, 2D convolutions, 3D convolutions, 4D convolutions, 5D convolutions, dilated or atrous convolutions, transpose convolutions, depthwise separable convolutions, pointwise convolutions, 1 x 1 convolutions, group convolutions, flattened convolutions, spatial and cross-channel convolutions, shuffled grouped convolutions, spatial separable convolutions, and deconvolutions. The neural network-based base caller 300 can use one or more loss functions such as logistic regression/log loss, multi-class cross-entropy/softmax loss, binary cross-entropy loss, mean-squared error loss, L1 loss, L2 loss, smooth L1 loss, and Huber loss. The neural network-based base caller 300 can use any parallelism, efficiency, and compression schemes such TFRecords, compressed encoding (*e.g.*, PNG), sharding, parallel calls for map transformation, batching, prefetching, model parallelism, data parallelism, and synchronous/asynchronous stochastic gradient descent (SGD). The neural network-based base caller 300 can include upsampling layers, downsampling layers, recurrent connections, gates and gated memory units (like an LSTM or GRU), residual blocks, residual connections, highway connections, skip connections, peephole connections, activation functions (*e.g.*, non-linear transformation functions like rectifying linear unit (ReLU), leaky ReLU, exponential liner unit (ELU), sigmoid and hyperbolic tangent (tanh)), batch normalization layers, regularization layers, dropout, pooling layers (*e.g.*, max or average pooling), global average pooling layers, and attention mechanisms.

[0080] The neural network-based base caller 300 is trained using backpropagation-based gradient update techniques. Example gradient descent techniques that can be used for training the neural network-based base caller 300 include stochastic gradient descent, batch gradient descent, and mini-batch gradient descent. Some examples of gradient descent optimization algorithms that can be used to train the neural network-based base caller 300 are Momentum, Nesterov accelerated gradient, Adagrad, Adadelata, RMSprop, Adam, AdaMax, Nadam, and AMSGrad.

[0081] In one implementation, the neural network-based base caller 300 uses a specialized architecture to segregate the processing of data for different sequencing cycles. The motivation for using the specialized architecture is described first. As discussed above, the neural network-based base caller 300 processes image patches for a current sequencing cycle, one or more preceding sequencing cycles, and one or more successive sequencing cycles. Data for additional sequencing cycles provide sequence-specific context. The neural network-based base caller 300 learns the sequence-specific context during training and base calls the sequencing cycles.

Furthermore, data for pre and post sequencing cycles provide second order contribution of pre-phasing and phasing signals to the current sequencing cycle.

[0082] However, images captured at different sequencing cycles and in different image channels are misaligned and have residual registration error with respect to each other. To account for this misalignment, the specialized architecture comprises spatial convolution layers that do not mix information between sequencing cycles and only mix information within a sequencing cycle.

[0083] Spatial convolution layers (or spatial logic) use so-called “segregated convolutions” that operationalize the segregation by independently processing data for each of a plurality of sequencing cycles through a “dedicated, non-shared” sequence of convolutions. The segregated convolutions convolve over data and resulting feature maps of only a given sequencing cycle, *i.e.*, intra-cycle, without convolving over data and resulting feature maps of any other sequencing cycle.

[0084] Consider, for example, that the input image data comprises (i) the current image patch for a current (time t) sequencing cycle to be base called, (ii) the previous image patch for a previous (time $t-1$) sequencing cycle, and (iii) the next image patch for a next (time $t+1$) sequencing cycle. The specialized architecture then initiates three separate convolution pipelines, namely, a current convolution pipeline, a previous convolution pipeline, and a next convolution pipeline. The current data processing pipeline receives as input the current image patch for the current (time t) sequencing cycle and independently processes it through a plurality of spatial convolution layers to produce a so-called “current spatially convolved representation” as the output of a final spatial convolution layer. The previous convolution pipeline receives as input the previous image patch for the previous (time $t-1$) sequencing cycle and independently processes it through the plurality of spatial convolution layers to produce a so-called “previous spatially convolved representation” as the output of the final spatial convolution layer. The next convolution pipeline receives as input the next image patch for the next (time $t+1$) sequencing cycle and independently processes it through the plurality of spatial convolution layers to produce a so-called “next spatially convolved representation” as the output of the final spatial convolution layer.

[0085] In some implementations, the current, previous, and next convolution pipelines are executed in parallel. In some implementations, the spatial convolution layers are part of a spatial convolution network (or subnetwork) within the specialized architecture.

[0086] The neural network-based base caller 300 further comprises temporal convolution layers (or temporal logic) that mix information between sequencing cycles, *i.e.*, inter-cycles. The temporal convolution layers receive their inputs from the spatial convolution network and

operate on the spatially convolved representations produced by the final spatial convolution layer for the respective data processing pipelines.

[0087] The inter-cycle operability freedom of the temporal convolution layers emanates from the fact that the misalignment property, which exists in the image data fed as input to the spatial convolution network, is purged out from the spatially convolved representations by the stack, or cascade, of segregated convolutions performed by the sequence of spatial convolution layers.

[0088] Temporal convolution layers use so-called “combinatory convolutions” that groupwise convolve over input channels in successive inputs on a sliding window basis. In one implementation, the successive inputs are successive outputs produced by a previous spatial convolution layer or a previous temporal convolution layer.

[0089] In some implementations, the temporal convolution layers are part of a temporal convolution network (or subnetwork) within the specialized architecture. The temporal convolution network receives its inputs from the spatial convolution network. In one implementation, a first temporal convolution layer of the temporal convolution network groupwise combines the spatially convolved representations between the sequencing cycles. In another implementation, subsequent temporal convolution layers of the temporal convolution network combine successive outputs of previous temporal convolution layers. The output of the final temporal convolution layer is fed to an output layer that produces an output. The output is used to base call one or more clusters at one or more sequencing cycles.

[0090] Figure 3 depicts a diagram of the neural network-based base caller 300 that can effectively process sequencing images generated during a sequencing process, in accordance with one implementation. Referring to Figure 3, the neural network-based base caller 300 can be executed using a system described herein in accordance with one exemplary implementation. It should be noted that the example shown in Figure 3 can be referred to as a five-cycle input, one-cycle output neural network. As such, the inputs to the multi-cycle neural network model include five (5) spatially aligned patches (*e.g.*, 301), for example, patches from the tile data arrays of five sensing cycles of a given tile. In the example, spatially aligned patches can have the same aligned row and column dimensions (x, y) as other patches in the set, so that the information relates to the same clusters of genetic material on the tile in sequence cycles. In addition, a subject patch is a patch from the array of tile data for cycle N, such that, for example, the set of five spatially aligned patches includes a patch from cycle N-2 preceding the subject patch (cycle N) by two cycles, a patch from cycle N-1 preceding the subject patch (cycle N) by one cycle, a patch from cycle N+1 following the patch from the subject cycle (cycle N) by one cycle, and a patch from cycle N+2 following the patch from the subject cycle (cycle N) by two cycles.

[0091] The model (or neural network-based base caller 300) depicted in Figure 3 also includes a segregated stack of multiple layers of the neural network for each of the input patches (e.g., segregated stacks 302). In other words, the segregated stack 302 receives as input tile data for the patch from cycle N-2 (301), and it is segregated from the stacks 304, 306, 308, and 310 that receive data from other cycles, so that the segregated stacks do not share their input data, or intermediate data, with each other.

[0092] In some implementations, all of the individual segregated stacks can have an identical (or virtually identical) model, and identical (or virtually identical) trained parameters. In other implementations, the models and trained parameters may vary (*i.e.*, can be different in the different stacks). In any case, as a segregated stack, stack 302 can receive, as input, tile data for the patch from cycle N-2. Stack 304 can receive, as input, tile data for the patch from cycle N-1. Stack 306 can receive, as input, tile data for the patch from cycle N. Stack 308 can receive, as input, tile data for the patch from cycle N+1. Stack 310 can receive, as input, tile data for the patch from cycle N+1.

[0093] Referring again to Figure 3, the layers of the segregated stacks (e.g., 312, 322, 332, 342, 352, 362, and 372) can each execute one or more convolution operation(s) of one or more kernels, including a plurality of filters over their respective input data for the layer. As an example, a patch associated with Cycle N-2 may include three (3) features as it is input to the first layer, namely, the layer 312. However, the output of the layer 312 may include many more features, for example, 10 to 20 features. Likewise, the outputs of each of layers 322, 332, 342, 352, 362, and 372 may include virtually any number of features suitable for a particular implementation. The parameters of the filters can be trained parameters for the neural network, for example, as weights and biases. The output feature set (intermediate data) from each of the stacks 302, 304, 306, and 310 is provided as input to an inverse hierarchy 382 of temporal combinatorial layers, in which the intermediate data from the multiple cycles are combined. In the example depicted in Figure 3, the inverse hierarchy 382 includes a first layer 384 and a second layer 386, such that the first layer includes three combinatorial layers, each receiving intermediate data from three of the segregated stacks, and the second (or final) layer includes one combinatorial layer receiving intermediate data from the three temporal layers.

[0094] The output of the second (or final) combinatorial layer 382 can be an output patch of classification data for clusters located in the corresponding patch of the tile from cycle N. In addition, the output patches can be assembled into an output array of classification data for the tile for cycle N. In some implementations, the output patch may have sizes and dimensions different from the input patches. In some implementations, the output patch may include pixel-by-pixel data that can be filtered by the host to select cluster data.

[0095] The output classification data can then be applied to a softmax function 394 (or other output activation function) optionally executed by the host, or on the configurable processor, depending on a particular implementation. An output function different from the softmax function 394 can also be used (*e.g.*, used for making a base call output parameter according to the largest output, then use a learned nonlinear mapping using context/network outputs to give base quality).

[0096] Finally, the output of the softmax function 394 can be provided as base call predictions 396 for the cycle N and stored in the host memory to be used in subsequent processing. Other systems may use another function for output probability calculation (*e.g.*, another nonlinear model).

[0097] The hierarchy 382 of temporal combinatorial layers 384, 386 can collectively represent a temporal logic of the neural network-based base caller 300, and the segregated stacks 302, 304, 306, 308, and 310 can collectively represent a spatial logic of the neural network-based base caller 300.

[0098] The neural network-based base caller 300 can, for example, be implemented using a configurable processor with a plurality of execution clusters, such that a complete evaluation of one tile cycle within the duration of the time interval, or close to the duration of the time interval, of one sensing cycle, can effectively provide the output data in real time. Data flow logic can be configured to distribute input units of tile data and trained parameters to the execution clusters, and to distribute output patches for aggregation in memory.

[0099] Additional details about the neural network-based base caller 300 can, for example, be found in US Provisional Patent Application No. 62/821,766, titled “ARTIFICIAL INTELLIGENCE-BASED SEQUENCING,” (Attorney Docket No. ILLM 1008-9/IP-1752-PRV), filed on March 21, 2019, which is incorporated herein by reference.

Index Images

[00100] Index images depict nucleotides with low-complexity patterns in which some of the four bases A, C, T, and G are represented at a frequency of less than 15%, 10%, or 5% of all the nucleotides. This is the case because, for any given index sequencing cycle, an index image depicts intensity emissions of (1) multiple analytes that originate from the same sample and share the same index sequence, and also of (2) analytes that belong to different samples and have different index sequences.

[00101] In other words, the first type of analytes has the same index base for every index sequencing cycle. As a result, the index image ends up depicting the same nucleotide for multiple analytes. This reduces the nucleotide diversity of the index image.

[00102] The index image's nucleotide diversity is further reduced when the second type of analytes also ends up having the same index base for certain index sequencing cycles. This happens for two reasons. First, the index sequences are short sequences with two to twenty index bases and thus do not have enough positions that can create significant mismatches between different index sequences. Second, often, up to only twenty samples are pooled for simultaneous sequencing. As a result, the number of different index sequences that can be depicted by an index image is not substantial. These factors lead to different index sequences having matching index bases at the same positions (base collision), which in turn causes the analytes with different index sequences to have the same index base for certain index sequencing cycles.

[00103] Low nucleotide diversity in the index images creates intensity patterns that lack signal diversity (contrast). On the other hand, the target images depict nucleotides with high-complexity patterns in which each of the four bases A, C, T, and G is represented at a frequency of at least 20%, 25%, or 30% of all the nucleotides. This is the case because the target sequences are often long (*e.g.*, one-fifty bases) and are unique to each analyte regardless of the source sample. Therefore, unlike the index images, the target images have adequate signal diversity.

[00104] Convolution kernels and filters of the neural network-based base caller 300 are trained largely on the target images. So, when, during inference, the trained neural network-based base caller 300 is presented with index images that have not undergone preprocessing (raw index images), its base calling accuracy for the index reads drops because its convolution kernels and filters are trained to detect intensity patterns based on the contrast.

[00105] Bypassing preprocessing by training the neural network-based base caller 300 on large amounts of raw index images to introduce signal diversity is not feasible because only so many index sequences are published and made publicly available. Furthermore, it is not uncommon for users to design custom index sequences and use them instead of the published index sequences. So, when trained on just the raw index images, the neural network-based base caller 300 may not be able to “generalize” well or effectively during inference and it may be prone to a problem that can be referred to as “overfitting.”

Improved Techniques

[00106] The improved techniques for base calling index images include reordering of sequencing images, and training of the neural network-based base caller 300 where a “temporal” logic (or component or layer) can be effectively “frozen” (or bypassed). In addition, the improved techniques include various combinations, including, for example, combining “normalization” of sequencing images with reordering of sequencing images and/or with effective “freezing” of the temporal logic.

[00107] In accordance with one aspect, a set of sequencing images can be reordered to generate one or more reordered sets of sequencing images. The set of sequencing images that are reordered (reordered sequencing images) can include index images of one or more indexes provided for one or more libraries used for multiplexing, in addition to target images associated with a target being analyzed (*e.g.*, template strand). In any case, the reordering of sequencing images can provide several additional sequencing images as additional training data for the training of the neural network-based base caller 300, as it is possible to generate multiple reordered sets of sequencing images based on a single set of sequencing images, for example, an image in its original consecutive sequencing cycle (*e.g.*, cycle 1, cycle 2...cycle M). For example, for a consecutive sequencing cycle starting from cycle 1, cycle 2...cycle 8, where $M=8$, it would be possible to generate about $(M^N - 1)$ or just about M^N sets of rearranged sequencing images, where N can, for example, represent a selection window size (*e.g.*, 3, 4, 5, 6, 7) for datasets provided as training data for the training of the neural network-based base caller 300 (*e.g.*, 8^5 , 8^6 , 8^7).

[00108] As such, the reordering of a set of sequencing images can be especially useful for index images given their relatively low diversity and/or limited availability. It should be noted that the reordering of the sequencing images can be accomplished in various ways. For example, in accordance with one or more implementations, a “shuffling” technique can be utilized to reorder a set of sequencing images by effectively maintaining one or more images fixed in a central position (*e.g.*, a third image in a sequence of five consecutively numbered images from one to five) while exchanging the position of one or more other images between left (*e.g.*, first image, second images) and right flanks (*e.g.*, fourth image, fifth image) relative to the central position. Other techniques for reordering sequencing images include, for example, permutations with or without repetition, combinations, etc. The techniques can be used to effectively generate the rearrangement of a set of sequencing images based on one or more parameters (*e.g.*, set size, selection window size) in accordance with additional implementations. Again, rearranging of the sequencing images can be especially useful for index images given their relatively lower level of diversity.

[00109] In addition to the reordering of sequencing images, in accordance with another aspect, the temporal logic of the neural network-based base caller 300 can be effectively “frozen” or bypassed to effectively prevent the temporal logic from learning temporal dependencies (*e.g.*, sequential, chronological, or consecutive patterns). In doing so, the temporal logic can be already trained to at least some extent (*e.g.*, previously trained on a first dataset). As such, “freezing” or bypassing of the temporal logic of a neural network can take place in the form of further (or additional) training using a second set of training data that trains the spatial logic but does not

train the temporal logic at least with respect to backward propagation (*e.g.*, weights update). “Freezing” or the bypassing of the temporal logic of a neural network can be especially useful for index images given their relatively lower level of diversity.

[00110] In accordance with yet another aspect, the neural network-based base caller 300 can be trained over multiple training iterations, such that in each iteration each one of the respective sequencing images is processed through the neural network-based base caller 300 to generate respective base call predictions. In doing so, one or more weights of the neural network-based base caller can be effectively improved (*e.g.*, via backpropagation) based on comparisons of the respective base call predictions against respective base call ground truths.

[00111] As noted above, in accordance with other aspects, improved techniques also include various combinations including, for example, combining the “normalization” of sequencing images with the reordering of sequencing images and/or with effective “freezing” of the temporal logic. These and other aspects and implementations will become even more apparent in view of the description below with accompanying Figures 4A-21. Since freezing is applied during the training of the neural network-based base caller 300, it can be combined with the reordering of sequencing images and/or the normalization during training. In other implementations, the trained neural network-based base caller 300 trained based on the freezing can use the reordering of sequencing images and/or the normalization during inference.

[00112] To further elaborate, Figure 4A depicts another multi-cycle neural network model (or neural network-based base caller 300) in accordance with another implementation. Referring to Figure 4A, the neural network-based base caller 300 can effectively process the sequencing images 406 generated during a sequencing process. It should be noted that the sequencing images 406 can include index images associated with one or more indexes of one or more libraries (*e.g.*, index A, index B shown in Figure 1) as well as target images associated with a target (*e.g.*, a template strand itself without an index).

[00113] In essence, the neural network-based base caller 300 can be executed to process the sequencing images 406 as input, for example, similar to or substantially like the processing described above with respect to the neural network-based base caller 300 (shown in Figure 3). Again, a five-cycle input, one-cycle output neural network is shown here as an example. As such, the inputs to the multi-cycle neural network model (or the neural network-based base caller 300) depicted in Figure 4A can include five spatially aligned patches from the tile data arrays of five sensing cycles of a given tile. However, for simplification, segregated stacks and other components of the neural network-based base caller 300 are not shown in Figure 4A (depicted in Figure 3). Instead, in Figure 4A, the sequencing images 406 are each depicted as having multiple datasets corresponding to multiple channels used to generate images in sequence.

[00114] In any case, referring to Figure 4A, the neural network-based base caller 300 is configured to access a set of sequencing images 406. It should be noted that the sequencing images 406 can be a set of sequencing images arranged in a consecutive order, for example, in the order that they were generated as an index or read in a sequencing cycle. In other words, sequencing images in the set of sequencing images 406 can be arranged in the order of sequencing cycles, N-2, N-1, N, N+1, N+2 (or cycles 1, 2, 3, 4 and 5) as shown in Figure 4A. As such, the neural network-based base caller 300 can process the sequencing images 406 as input, in a similar manner as described above with respect to the neural network-based base caller 300 (shown in Figure 3).

[00115] In processing the sequencing images 406 as input, the neural network-based base caller 300 can effectively perform training and/or can be trained for base calls (*i.e.*, training the neural network-based base caller 300) over a plurality of training iterations, such that each one of the respective training iterations processes its respective set of the sequencing images (406) through the neural network-based base caller 300 (through its stacks and/or its layers) to generate a respective base call prediction, for example, a base call 422 for cycle N (or cycle 3). In doing so, an error can be determined based on its associated ground truth. For example, an error 424 can be determined based on a cycle N (or cycle 3) ground truth 426. Based on the error 424, weights of the neural network-based base caller 300 can be updated via backward propagation 432.

[00116] Moreover, the neural network-based base caller 300 can also be configured to reorder the sequencing images 406 and/or process one or more reordered sequencing images as will be described in greater detail below (*e.g.*, rearranging sequencing images in an order that is different from the consecutive order of the images in the sequencing cycles used to generate the images). Generally, the sequencing images 406 can be reordered (or rearranged) from a first order to a second order, different than the first order, as will be described in greater detail below. For ease of illustration here, rather than a more general approach, a more specific approach, namely, “shuffling” is presented first before presenting more general approaches including, for example, “permutation” that may yield more reordering options.

Reordering by Shuffling

[00117] To further elaborate, Figure 4B depicts the neural network-based base caller 300 in connection with a reordered set of the sequencing images 446 in accordance with one implementation. Although it is not shown in Figure 4B, it should be noted that the neural network-based base caller 300 can also be configured to generate the reordered set of the sequencing images 446, by effectively reordering the sequencing images 406 before processing

the reordered set of the sequencing images 446 as input. Alternatively, the neural network-based base caller 300 can also be configured to merely process the reordered set of the sequencing images 446 as input without generating the reordered set of the sequencing images 446. For simplicity, the neural network-based base caller 300 is depicted herein to receive the reordered set of the sequencing images 446 as input, but it is apparent that it can also be configured to reorder a set of sequencing images in a first order (*e.g.*, sequencing images 406) to generate a reordered set of sequencing images in a second order, different than the first order (*e.g.*, sequencing images 446).

[00118] As shown in Figure 4B, the neural network-based base caller 300 can receive the reordered set of sequencing images 446 as input. In any case, the reordered set of the sequencing images 446 is a reordering of the set of the (original) sequencing images 406 representing a first order, namely: cycle 1, cycle 2, cycle 3, cycle 4 and cycle 5 (*e.g.*, the original sequencing cycles as the images are/were obtained). As such, the sequencing images of the reordered set of the sequencing images 446 are in a second order which is different than the first order. In the example shown in Figure 4B, the reordered set of the sequencing images 446 is depicted as having the following reordering: cycle 4, cycle 5, cycle 3, cycle 1, and cycle 2. As such, the reordering depicted in Figure 4A shows an example where an image in a central (cycle) position, namely cycle N, is kept fixed at its central position (cycle 3) while the images from left flanking cycles (N-2 and N-1) corresponding to cycle 1 and cycle 2 are effectively swapped with the images corresponding to cycle 4 and cycle 5 from the right flanking cycles (N+1 and N+2) relative to the fixed central position (cycle). This kind of reordering, swapping, or exchange can be referred to as “shuffling.”

[00119] More generally, in reordering by “shuffling” a set of sequencing images (*e.g.*, sequencing images 406), a particular one or more image(s) of a set of sequencing images can be kept at its position, and the position of one or more of the other sequencing images can be changed. In other words, at least one of the sequencing images at a right flanking position and/or a left flanking position can be changed between a first order of a first set of sequencing images (*e.g.*, sequencing images 406) and a second order (or reordering) of the same set sequencing images (*e.g.*, sequencing images 446). As another example, at least some of the sequencing images at one or more right flanking positions and/or one or more left flanking positions can be kept fixed between a first and a second order.

[00120] In processing the sequencing images 446 as input, the neural network-based base caller 300 can effectively perform training and/or can be trained for base calls (*i.e.*, training the neural network-based base caller 300) over a plurality of training iterations, such that each one of the respective training iterations processes its respective set of the sequencing images (446)

through the neural network-based base caller 300 (through its stacks and/or its layers) to generate a respective base call prediction, for example, a base call 452 for cycle N (or cycle 3). In doing so, an error can be determined based on its associated ground truth. For example, an error 454 can be determined based on a cycle N (or cycle 3) ground truth 426. Based on the error 454, weights of the neural network-based base caller 300 can be updated via backward propagation 462.

[00121] To provide another example, Figure 4C depicts another exemplary reordered set of the sequencing images 476 as “cycle 1, cycle 4, cycle 3, cycle 2, and cycle 5).” In processing the sequencing images 476 as input, the neural network-based base caller 300 can effectively perform training and/or can be trained for base calls (*i.e.*, training the neural network-based base caller 300) over a plurality of training iterations, such that each one of the respective training iterations processes its respective set of the sequencing images (476) through the neural network-based base caller 300 (through its stacks and/or its layers) to generate a respective base call prediction, for example, a base call 482 for cycle N (or cycle 3). In doing so, an error can be determined based on its associated ground truth. For example, an error 484 can be determined based on a cycle N (or cycle 3) ground truth 426. Based on the error 484, weights of the neural network-based base caller 300 can be updated via backward propagation 492.

[00122] Similarly, Figures 5A, 5B, and 5C depict yet other exemplary reordered sets of the sequencing images 506, 546, and 576, in accordance with yet other exemplary implementations, where an image in a central cycle (N) (or cycle 3) remains fixed while one or more other images of one or more cycles have been reordered. For example, in the reordered set of the sequencing images 506, cycle 2 and cycle 3 are both retained in their original positions, but in the reordered sets of the sequencing images 546 and 576, only cycle 3 remains fixed, where one or more other cycles may have been changed and/or swapped between left and right flanks relative to cycle 3 that remains fixed in its central position.

[00123] Referring to Figures 5A, 5B, and 5C, it should be noted that the neural network-based base caller 300 can process each one of the reordered (or shuffled) sets of sequencing images 506, 546, and 576, in a similar manner as described above with respect to the neural network-based base caller 300. In processing the reordered sets of sequencing images 506, 546, and 576, the neural network-based base caller 300 can effectively perform training or it can be trained for base calls (*i.e.*, the neural network-based base caller 300 can be trained) over a plurality of training iterations, such that each one of the respective training iterations processes its respective set of the reordered set of sequencing images 506, 546, and 576 through the neural network (not shown) in the neural network-based base caller 300 through its stacks and/or its layers (not shown) to generate a respective base call prediction, for example, a base call 522 for cycle N (or

cycle 3) as depicted in Figure 5A. In doing so, an error can be determined based on its associated ground truth. For example, an error 524 can be determined based on a cycle N (or cycle 3) ground truth 526, as also depicted in Figure 5A. As a result, one or more factors (*e.g.*, weights) can be updated via a process, generally known as backward propagation, as an important part of the training iterations. As an example, backward propagation 532 is depicted in Figure 5A. Other examples of errors 554 and 584 and corresponding backward propagations 562 and 592 are also depicted.

[00124] The examples of reordering noted above represent more specific exemplary implementations where, for example, one or more of the sequencing images can be determined to remain fixed in their original position (*e.g.*, a center position N is kept fixed as shown in Figures 4A, 4B, and 4C) or one or more sequencing images can be determined to be moved from their original location. However, it will be appreciated that a set of sequencing images can be virtually reordered in any way possible and/or desired, including, for example, by using random selection with or without replacement (or with or without repetition).

[00125] Moreover, it will also be appreciated that the reordering of a set of sequencing images into one or more reordered sets of sequencing images can be achieved through even more methodologies, in a manner that may even better address the needs of base calling using the neural network-based base caller 300. By way of example, a subset selector (*e.g.*, subset selector component, subset selector logic) can be configured to effectively generate a reordering of a set of sequencing images (reordered set of sequencing images) based on one or more parameters, as will be described below in accordance with yet other implementations.

Random Permutation

[00126] To further elaborate, Figure 6 depicts a subset selector 612 (*e.g.*, subset selector component, subset selector logic) configured to effectively generate a reordering of a set of sequencing images (reordered set of sequencing images) as a reordered subset of sequencing images, based on two (2) parameters in accordance with one implementation. Referring to Figure 6, a first parameter (M) represents a set of sequencing images (*e.g.*, a complete set of sequencing images that can be generated in a sequencing run, which currently can be eight (8) or ten (10) in some existing implementations). For simplicity, in Figure 6, the first parameter (M) is depicted to be eight (8) representing eight (8) consecutive cycles (cycle 1, cycle 2... cycle 8) 602. In addition, in Figure 6, a second parameter (N) is depicted to be five (5), six (6) or seven (7). Generally, the second parameter (N) can represent the size of a subset for selection from the set of M images. As such, the second parameter (N) can, for example, represent a “window” for selection of the input to the neural network-based base caller 300. Although not shown, it should

be noted that the subset selector 612 can, for example, be provided by the neural network-based base caller 300 (shown in Figure 4). Of course, the subset selector 612 can also be provided as a separate entity configured as it will be readily understood by those skilled in the art. The reordering logic implemented by the subset selector 612 can be random or driven by sequentially progressing to different permutations and combinations, either until exhaustion or up till early exit due to satisfaction of a terminal condition.

[00127] In any case, as depicted in Figure 6, the subset selector 612 can effectively obtain (*e.g.*, determine, identify, receive as input) the set of M, or eight consecutive cycles (ordered consecutively from cycle 1-cycle 8) 602 in the example and generate various subsets of N, for example, five (5) cycles in various reordered subsets as reordered subsets of sequencing images 626, 636, 646, and so on. As the exemplary rearranged subset of sequencing images 626 shows, there can be repetition of cycles, for example, cycle 8 (or more accurately the images in cycle 8) can be repeated in the reordered subset of sequencing images 626 as generated by the subset selector 612.

[00128] As also depicted in Figure 6, the reordered subsets of sequencing images 656 and 666 respectively depict examples of subsets where N is six (6) and seven (7). Generally, for the set of M elements (*e.g.*, eight (8)), about M to the power of N (M^N) (*e.g.*, 8^5) subsets can be determined when there is repetition (*i.e.*, the same cycle can be repeated). The number of possible subsets can be slightly less if no repetition is allowed and/or desired, as it will be readily known to those skilled in the art. As those skilled in the art will also readily appreciate, various “permutations” and “combinations” can be selected from an original set of sequencing images (*e.g.*, sequencing images 602 with cycles 1-cycle 8) based on one or more parameters (*e.g.*, M and N parameters).

[00129] Generally, multiple subsets of N sequencing images can be selected from a set of M sequencing images, where $M > N$. In other words, there can be multiple images in a single cycle, where each one of the multiple images can, for example, correspond to a single channel, as for example, two channels are depicted for the sequencing images shown in Figure 6. It should also be noted that respective subsets of the multiple selected subsets can arrange the N sequencing images in respective sequences of the N sequencing images. Also, the respective sequences can have different combinations of N sequencing images (*i.e.*, the combinations of the N sequencing images can vary from sequence-to-sequence). In addition, the combinations can be combinations with or without replacement. Similarly, the permutations can be permutations with or without replacement. The respective sequences can arrange the N sequencing images in accordance with consecutive or non-consecutive sequencing cycles. It should also be noted that N can be determined to be a constant for the selection of respective subsets of the multiple selected

subsets, or N can be selected to be a variable, so that N may vary between the respective subsets of the multiple selected subsets. Furthermore, the respective subsets of the multiple selected subsets can be selected to have or not have any overlapping sequencing images.

[00130] Training of the neural network-based base caller 300 can be achieved over multiple training iterations, where each one of the training iterations processes its respective sequences of the N sequencing images through the neural network-based base caller 300 to generate respective base call predictions, and update the weights (*e.g.*, via backpropagation) of the neural network-based base caller 300, based on the comparison of the respective base call predictions against the respective base call ground truths.

[00131] As noted above, the subset selector 612 (shown in Figure 6) can, for example, be provided by the neural network-based base caller 300 (shown in Figure 4) or as a separate entity. However, for simplicity, additional implementations depict the neural network-based base caller 300 as receiving already reordered (or rearranged) sequencing images. The depicted reordered sequencing images can, for example, be generated by the subset selector 612 (shown in Figure 6).

[00132] More specifically, in accordance with one implementation, Figure 7A depicts a reordered subset of sequencing images 626 (also shown in Figure 6) as it can, for example, be generated by the subset selector 612 (shown in Figure 6) in connection with the neural network-based base caller 300 (shown in Figure 4). Referring to Figure 7A, the neural network-based base caller 300 can receive, as input, the reordered subset of sequencing images 626. The neural network-based base caller 300 can then process the reordered subset of sequencing images 626, in a similar manner as discussed above, to generate one or more base call predictions 702, while utilizing the base call ground truths 706 in a backward propagation 712, in consideration of an error 704.

[00133] Similarly, Figures 7B and 7C depict the neural network-based base caller 300 in connection with the reordered subsets of sequencing images 636 and 646, respectively, where base call predictions 722 and 742 can be made, while utilizing the base call ground truths 726 and 746 in backward propagations 732 and 752, in consideration of errors 724 and 744, in a similar manner as discussed above. Although not shown, additional reordered subsets of sequencing images can be provided to the neural network-based base caller 300 and processed by the neural network-based base caller 300 for the effective training of its neural network for base calling. In this way, the neural network-based base caller 300 can be trained over a number of training iterations, where each one of the training iterations processes its respective reordered sequences of N (*e.g.*, five) sequencing images through the neural network-based base caller 300 to generate respective base call predictions, and to update the weights (*e.g.*, via backpropagation)

of the neural network-based base caller 300, based on the comparison of the respective base call predictions against the respective base call ground truths.

[00134] To elaborate even further, Figures 8A and 8B depict additional examples of reordered subsets of sequencing images 656 and 666 (also shown in Figure 6) in connection with the neural network-based base caller 300, where the reordered subsets of sequencing images 656 and 666 can also serve in the training iterations process, alone or in combination with other training iterations, for example, in combination with training iterations using reordered subsets of sequencing images 626, 636, and 646 (shown in Figures 7A, 7B, and 7C) in accordance with additional implementations. As discussed above, the training will include determining errors 804 and 824 based on comparing base calling predictions 802 and 822 against base call ground truths 806 and 826 and executing back propagation iterations 812 and 832 based on the errors 804 and 824, respectively.

Bypassing Retraining of the Trained Temporal Logic

[00135] As noted above and discussed with respect to a few exemplary implementations, reordered sets of sequencing images can be generated in accordance with one implementation of the technology disclosed. In accordance with another aspect, the neural network-based base caller 300 can be trained (or further trained) by only utilizing the spatial logic of the (or previously trained) neural network-based base caller 300. In other words, the temporal logic of the neural network-based base caller 300 can be effectively “frozen” or bypassed in a second (or further) training where only the spatial logic (not the temporal logic) is trained at least with respect to backward propagation (e.g., weights update). A different (or second) training dataset can be used in the second (or further) training phase, after the initial (or first) training has been completed based on an initial (or first) training dataset.

[00136] A temporal logic can be provided, in addition to the spatial logic in the neural network-based base caller 300. Referring back to Figure 3, the hierarchy 382 of the temporal combinatorial layers 382 can represent a temporal logic of the neural network-based base caller 300, and the segregated stacks 302, 304, 306, 308 and 310 can collectively represent the spatial logic of the neural network-based base caller 300.

[00137] To elaborate further, Figure 9 depicts an initial (first) training 902 and a further (second) training 932 in accordance with two implementations. Referring to Figure 9, the initial training 902 can be used to effectively train the neural network-based base caller 300 (also shown in Figure 4) for making base call predictions. In doing so, the neural network-based base caller 300 can receive a first training dataset 912 and process it through two of its components, namely, a spatial logic 914 and a temporal logic 916. As noted above, the temporal logic 916

can, for example, be provided as the temporal combinatorial layers 382 (depicted in Figure 3), and the spatial logic 914 can, for example, be provided as the segregated stacks 302, 304, 306, 308 and 310 (also depicted in Figure 3) for the neural network-based base caller 300 depicted in Figure 9.

[00138] In processing the first training dataset 912 first through the spatial logic 914 and then through the temporal logic 916, the neural network-based base caller 300 can utilize backward propagation to update weights 926 and 928 associated with the spatial logic 914 and the temporal logic 916, respectively. In other words, both the spatial logic 914 and the temporal logic 916 can be utilized to make updates via backward propagation, in a similar manner as discussed above. It should be noted that the first training dataset 912 can be an ordered set of sequencing images or a set of reordered (reordered set) sequencing images, generated in a similar manner as discussed above. However, it may be desirable for at least some applications to initially use an ordered set of sequencing images in an original consecutive order of sequencing images (cycle 1, cycle 2, ... cycle M) in the initial training 902 before a further training 932 (or second training) takes place.

[00139] In any case, after the initial (or first) training 902 has been completed, a further (or second) training 932 can be performed. Referring to Figure 9, the further (or second) training 932 can be performed by the neural network-based base caller 300. In doing so, the neural network-based base caller 300 can receive and process the second training dataset 942 through two of its components, namely, a trained spatial logic 944 and a trained temporal logic 946. In processing the second training dataset 942, the neural network-based base caller 300 can utilize backward propagation to update weights 956 only of the trained spatial logic 944. That is, during the further (or second) training 932, the backward propagation does not update the weights of the trained temporal logic 946 to effectively “freeze” or bypass the trained temporal logic 946. In other words, backward propagation of gradient updates through the trained temporal logic 946 is bypassed during the further (or second) training 932. This means that in the further (or second) training 932, backward propagation is performed only through the trained spatial logic 944, where gradient updates are determined based on comparisons of the base call predictions against the base call ground truths only for the trained spatial logic 944 (not for the trained temporal logic 946).

[00140] Referring to Figure 9, the second training dataset 942 can be an ordered set of sequencing images, as originally obtained during sequencing, or a set of reordered (reordered set) sequencing images generated in a similar manner as discussed above. For example, the sequencing images in the first training dataset 912 can include read sequencing images generated for read sequencing cycles of a sequencing run. As another example, the second training dataset

942 can include index sequencing images generated for index sequencing cycles of the sequencing run. In the example, the index sequencing images can be reordered sequencing images or sequencing images obtained in a consecutive order. As yet another example, the sequencing images in the second training dataset 942 can include a combination of read sequencing images and index sequencing images.

[00141] Referring to Figure 9, it should be noted that the further (or second) training 932 can effectively retrain the trained spatial logic 944 to provide it as a retrained spatial logic (not shown). The retrained spatial logic can be different than the trained spatial logic 944 in its original state prior to the inception of the further (or second) training 932, while the trained temporal logic 946 can be effectively “frozen” to remain unchanged after the further (or second) training 932. After the further (or second) training 932 has completed, the combination of the retrained spatial logic (not shown) and the trained temporal logic 946 can be applied to or can be used at inference to generate new base call predictions for a set of (new) sequencing images. It will be appreciated that the bypassing of the backward propagation updating (*i.e.*, retraining) for the trained temporal logic 946 can prevent the neural network-based base caller 300 from learning temporal dependencies (*e.g.*, sequential, chronological, or consecutive patterns) between intensity features of the index sequencing images. Learning temporal dependencies can hinder the effectiveness of training for calling index images at inference. As a result of avoiding the learning of temporal dependencies during the further (or second) training 932, while allowing the trained spatial logic 944 to be further trained by using backward propagation to provide the weights update 956, even more accurate base calls can be achieved during inference.

[00142] To elaborate even further, Figure 10 depicts a method 1000 for training a neural network-based base caller in accordance with one implementation. The method 1000 can, for example, be used by the neural network-based base caller 300 (shown in Figure 9) to train its neural network. It should be noted that the neural network-based base caller can include a spatial logic configured to extract intensity features confined to per-cycle sequencing images in an input, and a temporal logic configured to extract intensity features spanning pan-cycle sequencing images in the input. Referring to Figure 10, initially, the spatial logic and the temporal logic of the neural network-based base caller are trained (1002) on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic. Next, further (or additional) training is performed (1012) using a second training dataset by further training only the spatial logic of the neural network-based base caller on the second training dataset of sequencing images to generate a retrained spatial logic. It should be noted that the further training (1012) can comprise the forward propagating of sequencing images in the second training dataset through the trained spatial logic and the trained temporal logic and

generating further base call predictions. However, in the further training (1012) backward propagating is limited only through the trained spatial logic (not the temporal logic) such that further gradient updates are determined based on the comparison of the further (or additional) base call predictions against the further base call ground truths, thereby bypassing backward propagating the further gradient updates through the trained temporal logic. After the further training (1012), only the trained spatial logic of the neural network-based base caller is retrained (further trained) on a second training dataset of sequencing images to generate a retrained/further/additionally trained spatial logic. The method 1000 can proceed to apply (1022) the retrained spatial logic and the trained temporal logic of the neural network-based base caller to generate new base call predictions for (or on) new sequencing images at or during inference where, for example, base calls can be made for a target sample or template strand with an added index. The method 1000 can end after new base call predictions have been generated at or during inference. For example, the sequencing images in the first training dataset can include read sequencing images generated for read sequencing cycles of a sequencing run, as noted above. As another example, the second training dataset can include index sequencing images generated for index sequencing cycles of the sequencing run. As yet another example, the sequencing images in the second training dataset can include a combination of the read sequencing images and index sequencing images.

[00143] To elaborate still further, Figure 11 depicts another method 1100 for training a neural network-based base caller in accordance with another implementation. The method 1100 can, for example, be used by the neural network-based base caller 300 to train its neural network. The method 1000 can proceed in a similar manner as discussed above with respect to the method 1000 (shown in Figure 10) to train (1102) the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic. In addition, further (or additional) training can be performed (1112) on a second training dataset by further training only the spatial logic of the neural network-based base caller on the second training dataset of sequencing images to generate a retrained spatial logic, in a similar manner as discussed above. However, as depicted in Figure 11, the method 1100 can configure (1122) the neural network-based base caller with the retrained spatial logic and the trained temporal logic, thereby allowing future base calls (*e.g.*, during inference) to be executed (or be configured to execute or be configured for the purpose of executing) based on the neural network-based base caller with the retrained spatial logic.

[00144] More generally, a neural network-based base caller can be trained for the base calling of sequencing images in accordance with one aspect. To further elaborate, Figure 12 depicts a method 1200 for training a neural network-based base caller for the base calling of sequencing

images, in accordance with one implementation. The method 1200 can, for example, be used by the neural network-based base caller 300 to train its neural network. Referring to Figure 12, initially, multiple sequences of N sequencing images are obtained 1202 (*e.g.*, accessed, received, determined, identified). Thereafter, a neural network-based base caller is trained (1212) over multiple training iterations, such that in each one of the iterations, respective sequences of the N sequencing images are processed through the neural network-based base caller to generate respective base call predictions. In doing so, one or more weights of the neural network-based base caller can be effectively improved (*e.g.*, via backpropagation) for example, based on comparisons of the respective base call predictions against the respective base call ground truths.

[00145] To elaborate further yet, Figure 13 depicts another method 1300 for training a neural network-based base caller in accordance with one implementation. The method 1300 can, for example, be used by the neural network-based base caller 300 to train its neural network. Referring to Figure 13, initially, a set of sequencing images are accessed (1302). It should be noted that the sequencing images in the set of sequencing images are arranged in a first order. Next, the sequencing images are reordered 1312 (*e.g.*, randomly shuffled, permuted, one or more images are kept fixed in their original location while one or more other images are relocated from their original location in the sequence) in one or more reordered sets of sequencing images, such that the sequencing images in a reordered set are arranged in a second order that is different from the first order. After the reordering 1312 of the sequencing images from a first order to a second order, the reordered set of the sequencing images is processed (1322) to generate at least one base call prediction. Finally, a neural network-based base caller can be trained (1332) over multiple training iterations, such that each one of the respective training iterations processes a respective reordered set of the sequencing images through the neural network-based base caller to generate a respective base call prediction. During the training (1332) over multiple training iterations, each one of the respective training iterations can also improve the weights (*e.g.*, via backpropagation) of the neural network-based base caller based on one or more comparisons of the respective base call predictions against the respective base call ground truths.

[00146] It should be noted that the techniques and implementations described above can be combined with other techniques and implementations including the improved techniques described here, as well as other techniques (*e.g.*, normalization of index images). For example, one or more techniques involving the reordering of sequencing images (*e.g.*, shuffling, permutation) and effectively “freezing” a temporal logic (or layer) or bypassing the trained temporal logic that have been described above can be combined with one or more normalization techniques, for example, by using one or more of the normalization techniques provided in U.S.

Patent Application No. 62/979,384, titled “ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed 20 February 2020 (Attorney Docket No. ILLM 1015-1/IP-1857-PRV), which has been incorporated by reference herein in its entirety and for all purposes. For example, in accordance with one implementation, an index image from a current index sequencing cycle can be normalized based on one or more of the following: (i) intensity values of index images from one or more preceding index sequencing cycles, (ii) intensity values of index images from one or more succeeding index sequencing cycles, and (iii) intensity values of index images from the current index sequencing cycle.

[00147] Figure 14 shows a flow chart 1400 of a system that executes the disclosed base calling, in accordance with one implementation. An accessing logic is configured to access a set of sequencing images at step 1402. Sequencing images in the set of sequencing images are arranged in a first order. A reordering logic is configured to reorder the sequencing images in a reordered set of the sequencing images at step 1412. The sequencing images in the reordered set of the sequencing images are arranged in a second order. The first order is different than the second order. A processing logic is configured to process the reordered set of the sequencing images and generate at least one base call prediction 1422.

Normalization of Index Images

[00148] Generally, normalization across index sequencing cycles also includes normalization across image channels within the image data of the index sequencing cycles. For example, consider three index sequencing cycles: a first index sequencing cycle, a second index sequencing cycle, and a third index sequencing cycle. Also consider that each of the first, second, and third index sequencing cycles has two index images: a first index image (*e.g.*, red index image) in a first image channel (*e.g.*, red channel) and a second index image (*e.g.*, green index image) in a second image channel (*e.g.*, green channel). A red index image from the second index sequencing cycle is normalized based on (i) the intensity values of red and green images from the first index sequencing cycle, (ii) the intensity values of red and green images from the third index sequencing cycle, and (iii) the intensity values of red and green images from the second index sequencing cycle. A green index image from the second index sequencing cycle is normalized based on (i) the intensity values of red and green images from the first index sequencing cycle, (ii) the intensity values of red and green images from the third index sequencing cycle, and (iii) the intensity values of red and green images from the second index sequencing cycle. In other implementations, the images can be in blue and violet color channels instead of or in addition to the red and green channels. Additional details for “normalization” techniques are provided, for example, in U.S. Patent Application No. 62/979,384, titled

“ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed 20 February 2020 (Attorney Docket No. ILLM 1015-1/IP-1857-PRV), which has been incorporated by reference herein in its entirety and for all purposes.

[00149] Furthermore, normalization can include index images from flanking index sequencing cycles because taken together, nucleotides depicted by the index images from the current, preceding, and succeeding index sequencing cycles are cumulatively more diverse than nucleotides depicted only by the index images from the current index sequencing cycle. Expanding the normalization to index images from the flanking index sequencing cycles also includes at least one index image from the preceding and/or succeeding index sequencing cycles that depicts one or more nucleotides in a detectable signal state.

[00150] It should be noted that normalization of sequencing images (*e.g.*, index images) can be performed either prior to reordering of the sequencing images or after they have been reordered. Similarly, sequencing images can be normalized for the first and/or second datasets of sequencing images, selectively, or as a complete set, during the first and/or second training phases of the neural network-based base caller 300. For example, one or more index images provided in a second training dataset in the further (or second) training of the neural network can be normalized in combination with effectively “freezing” the temporal logic of the neural network, in accordance with one exemplary implementation.

Augmentation of Index Images

[00151] To further elaborate, the exemplary implementation noted above and another implementation involving the combination of normalization and reordering of sequencing images are discussed below. In addition, measured comparisons with other techniques for training the neural network-based base caller 300 (*e.g.*, “augmentation”) are discussed too. For example, one implementation of preprocessing that uses augmentation can be used, where an image augmenter preprocesses the index images and the target images using an augmentation function. In the example, an image augmenter can multiply the intensity values of the index images and the target images with a scaling factor and add an offset value to the multiplication’s result. In another exemplary implementation, the image augmenter changes the contrast of the index images and the target images. In yet another exemplary implementation, the image augmenter changes the focus of the index images and the target images. Additional details for “augmentation” techniques are provided, for example, in U.S. Patent Application No. 62/979,384, titled “ARTIFICIAL INTELLIGENCE-BASED BASE CALLING OF INDEX SEQUENCES,” filed 20 February 2020 (Attorney Docket No. ILLM 1015-1/IP-1857-PRV), which has been incorporated by reference herein in its entirety and for all purposes.

Technical Effect and Performance Results as Objective Indicia of Inventiveness

[00152] Illumina Enrichment kits (DNA and RNA based) help to isolate and enrich specific regions of interest in a genome or transcriptome for sequencing. For example, Illumina DNA Prep with Enrichment (formerly known as Nextera Flex for Enrichment) with the TruSight Cancer Panel targets 94 genes and 284 SNPs that are associated with various cancers. The following describes the principles of an enrichment reaction, and the commonly used terminology when describing Illumina enrichment kits. With enrichment kits, ‘Reaction’ (abbreviated “rxn”) refers to the number of enrichment reactions that are provided with the kit. ‘Plexity’ (abbreviated “plex”) refers to the number of pre-enriched libraries that are pooled together in an enrichment reaction. For example, an 8-rxn x 12-plex kit has enough reagents to perform eight enrichments. In each enrichment reaction, 12 libraries can be pooled. Therefore, the total number of samples that can be prepared with this kit is 96 samples. The following figure illustrates how three libraries are pooled in one enrichment reaction. Figure 23 depicts an example of 96 indexes.

[00153] Figure 15 shows measured base calling error rates for the previously disclosed “augmentation” and “normalization” implementations of the neural network-based base caller 300, as applied on 1-Plex sequencing data. The results show that when the neural network-based base caller 300 is trained with index cycles using single cycle normalization (red), the error rates for certain cycles are really high. Blue and green curves show implementations of the neural network-based base caller 300 using augmentation and no normalization. They show reasonable performance but would be harder to implement because of additional floating point complexity. This shows the need for a better solution.

[00154] Figures 16A, B and C compare measured base calling error rates for three different implementations of the neural network-based base caller 300, as applied on different complexities (*e.g.*, 96-Plex, 1-Plex, and 3-Plex). The first implementation (in red) is the currently disclosed “reordering” or “shuffling” implementation of the neural network-based base caller 300 using the joint training technique discussed above. The second implementation (in blue) is the currently disclosed “freeze” implementation of the neural network-based base caller 300 using the joint training technique discussed above. The third implementation (in green) is the previously disclosed “augmentation with no normalization” implementation of the neural network-based base caller 300. As shown by the performance charts, both the currently disclosed implementations outperform the previously disclosed implementation.

[00155] Figures 17A, B and C compare measured base calling error rates for three different implementations of the neural network-based base caller 300, as applied on different complexities (*e.g.*, 4-Plex, 2-Plex, and 1-Plex). The first implementation (in red) is the currently

disclosed “reordering” or “shuffling” implementation of the neural network-based base caller 300 using the joint training technique discussed above. The second implementation (in green) is the currently disclosed “freeze” implementation of the neural network-based base caller 300 using the joint training technique discussed above. The third implementation (in blue) is the currently disclosed “random permutation” implementation (Figure 6) of the neural network-based base caller 300 using the joint training technique discussed above. As shown by the performance charts, all three of the currently disclosed implementations have comparable performance, with the random permutation showing slightly better results.

[00156] Figure 18 depicts a plot that compares the “demultiplex” results for tiles from runs with different Plexities for RTA (blue), a previously disclosed deep learning implementation (orange) (DL 3c Aug), and the currently disclosed joint training and shuffling implementation (green) (DL 5c Norm*). The results from the new implementation are especially better for 1-Plex runs when compared to both the RTA and the previously disclosed deep learning implementation. Demultiplex refers to what percentage of the base called index sequences can be resolved to a unique index with one allowed mismatch.

Sequencing System

[00157] Figures 19A and 19B depict one implementation of a sequencing system 1900A. The sequencing system 1900A comprises a configurable processor 1946. The configurable processor 1946 implements the base calling techniques disclosed herein. The sequencing system is also referred to as a “sequencer.”

[00158] The sequencing system 1900A can operate to obtain any information or data that relates to at least one instance of a biological or chemical substance. In some implementations, the sequencing system 1900A is a workstation that may be similar to a bench-top device or desktop computer. For example, a majority (or all) of the systems and components for conducting the desired reactions can be within a common housing 1902.

[00159] In particular implementations, the sequencing system 1900A is a nucleic acid sequencing system configured for various applications, including but not limited to de novo sequencing, resequencing of whole genomes or target genomic regions, and metagenomics. The sequencer may also be used for DNA or RNA analysis. In some implementations, the sequencing system 1900A may also be configured to generate reaction sites in a biosensor. For example, the sequencing system 1900A may be configured to receive a sample and generate surface attached clusters of clonally amplified nucleic acids derived from the sample. Each cluster may constitute or be part of a reaction site in the biosensor.

[00160] The exemplary sequencing system 1900A may include a system receptacle or interface 1910 that is configured to interact with a biosensor 1912 to perform desired reactions within the biosensor 1912. In the following description with respect to Figure 19A, the biosensor 1912 is loaded into the system receptacle 1910. However, it is understood that a cartridge that includes the biosensor 1912 may be inserted into the system receptacle 1910 and in some states the cartridge can be removed temporarily or permanently. As described above, the cartridge may include, among other things, fluidic control and fluidic storage components.

[00161] In particular implementations, the sequencing system 1900A is configured to perform a large number of parallel reactions within the biosensor 1912. The biosensor 1912 includes one or more reaction sites where desired reactions can occur. The reaction sites may be, for example, immobilized to a solid surface of the biosensor or immobilized to beads (or other movable substrates) that are located within corresponding reaction chambers of the biosensor. The reaction sites can include, for example, clusters of clonally amplified nucleic acids. The biosensor 1912 may include a solid-state imaging device (*e.g.*, CCD or CMOS imager) and a flow cell mounted thereto. The flow cell may include one or more flow channels that receive a solution from the sequencing system 1900A and direct the solution toward the reaction sites. Optionally, the biosensor 1912 can be configured to engage a thermal element for transferring thermal energy into or out of the flow channel.

[00162] The sequencing system 1900A may include various components, assemblies, and systems (or sub-systems) that interact with each other to perform a predetermined method or assay protocol for biological or chemical analysis. For example, the sequencing system 1900A includes a system controller 1906 that may communicate with the various components, assemblies, and sub-systems of the sequencing system 1900A and also the biosensor 1912. For example, in addition to the system receptacle 1910, the sequencing system 1900A may also include a fluidic control system 1908 to control the flow of fluid throughout a fluid network of the sequencing system 1900A and the biosensor 1912; a fluid storage system 1914 that is configured to hold all fluids (*e.g.*, gas or liquids) that may be used by the bioassay system; a temperature control system 1904 that may regulate the temperature of the fluid in the fluid network, the fluid storage system 1914, and/or the biosensor 1912; and an illumination system 1916 that is configured to illuminate the biosensor 1912. As described above, if a cartridge having the biosensor 1912 is loaded into the system receptacle 1910, the cartridge may also include fluidic control and fluidic storage components.

[00163] Also shown, the sequencing system 1900A may include a user interface 1918 that interacts with the user. For example, the user interface 1918 may include a display 1920 to display or request information from a user and a user input device 1922 to receive user inputs. In

some implementations, the display 1920 and the user input device 1922 are the same device. For example, the user interface 1918 may include a touch-sensitive display configured to detect the presence of an individual's touch, and it can also be configured to identify a location of the touch on the display. However, other user input devices 1922 may be used, such as a mouse, touchpad, keyboard, keypad, handheld scanner, voice-recognition system, motion-recognition system, and the like. As will be discussed in greater detail below, the sequencing system 1900A may communicate with various components, including the biosensor 1912 (*e.g.*, in the form of a cartridge), to perform the desired reactions. The sequencing system 1900A may also be configured to analyze data obtained from the biosensor to provide a user with desired information.

[00164] The system controller 1906 may include any processor-based or microprocessor-based system, including systems using microcontrollers, reduced instruction set computers (RISC), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), coarse-grained reconfigurable architectures (CGRAs), logic circuits, and any other circuit or processor capable of executing functions described herein. The above examples are exemplary only and are thus not intended to limit in any way the definition and/or meaning of the term system controller. In the exemplary implementation, the system controller 1906 executes a set of instructions that are stored in one or more storage elements, memories, or modules in order to do at least one of the following: obtain and/or analyze detection data. Detection data can include a plurality of sequences of pixel signals, such that a sequence of pixel signals from each of the millions of sensors (or pixels) can be detected over many base calling cycles. Storage elements may be in the form of information sources or physical memory elements within the sequencing system 1900A.

[00165] The set of instructions may include various commands that instruct the sequencing system 1900A or biosensor 1912 to perform specific operations such as the methods and processes of the various implementations described herein. The set of instructions may be in the form of a software program, which may form part of a tangible, non-transitory computer readable medium or media. As used herein, the terms "software" and "firmware" are interchangeable and include any computer program stored in memory for execution by a computer, including RAM memory, ROM memory, EPROM memory, EEPROM memory, and non-volatile RAM (NVRAM) memory. The above memory types are exemplary only and are thus not limiting as to the types of memory usable for storage of a computer program.

[00166] The software may be in various forms such as system software or application software. Further, the software may be in the form of a collection of separate programs, or a program module within a larger program or a portion of a program module. The software also

may include modular programming in the form of object-oriented programming. After obtaining the detection data, the detection data may be automatically processed by the sequencing system 1900A, processed in response to user inputs, or processed in response to a request made by another processing machine (*e.g.*, a remote request through a communication link). In the illustrated implementation, the system controller 1906 includes an analysis module 1944. In other implementations, the system controller 1906 does not include the analysis module 1944 and instead has access to the analysis module 1944 (*e.g.*, the analysis module 1944 may be separately hosted on the cloud).

[00167] The system controller 1906 may be connected to the biosensor 1912 and the other components of the sequencing system 1900A via communication links. The system controller 1906 may also be communicatively connected to off-site systems or servers. The communication links may be hardwired, corded, or wireless. The system controller 1906 may receive user inputs or commands, from the user interface 1918 and the user input device 1922.

[00168] The fluidic control system 1908 includes a fluid network and is configured to direct and regulate the flow of one or more fluids through the fluid network. The fluid network may be in fluid communication with the biosensor 1912 and the fluid storage system 1914. For example, select fluids may be drawn from the fluid storage system 1914 and directed to the biosensor 1912 in a controlled manner, or the fluids may be drawn from the biosensor 1912 and directed toward, for example, a waste reservoir in the fluid storage system 1914. Although not shown, the fluidic control system 1908 may include flow sensors that detect a flow rate or pressure of the fluids within the fluid network. The sensors may communicate with the system controller 1906.

[00169] The temperature control system 1904 is configured to regulate the temperature of fluids at different regions of the fluid network, the fluid storage system 1914, and/or the biosensor 1912. For example, the temperature control system 1904 may include a thermocycler that interfaces with the biosensor 1912 and controls the temperature of the fluid that flows along the reaction sites in the biosensor 1912. The temperature control system 1904 may also regulate the temperature of solid elements or components of the sequencing system 1900A or the biosensor 1912. Although not shown, the temperature control system 1904 may include sensors to detect the temperature of the fluid or other components. The sensors may communicate with the system controller 1906.

[00170] The fluid storage system 1914 is in fluid communication with the biosensor 1912 and may store various reaction components or reactants that are used to conduct the desired reactions therein. The fluid storage system 1914 may also store fluids for washing or cleaning the fluid network and biosensor 1912 and for diluting the reactants. For example, the fluid storage system 1914 may include various reservoirs to store samples, reagents, enzymes, other biomolecules,

buffer solutions, aqueous and non-polar solutions, and the like. Furthermore, the fluid storage system 1914 may also include waste reservoirs for receiving waste products from the biosensor 1912. In implementations that include a cartridge, the cartridge may include one or more of a fluid storage system, fluidic control system or temperature control system. Accordingly, one or more of the components set forth herein as relating to those systems can be contained within a cartridge housing. For example, a cartridge can have various reservoirs to store samples, reagents, enzymes, other biomolecules, buffer solutions, aqueous and non-polar solutions, waste, and the like. As such, one or more of a fluid storage system, fluidic control system or temperature control system can be engaged in a removable way with a bioassay system via a cartridge or other biosensor.

[00171] The illumination system 1916 may include a light source (*e.g.*, one or more LEDs) and a plurality of optical components to illuminate the biosensor. Examples of light sources may include lasers, arc lamps, LEDs, or laser diodes. The optical components may be, for example, reflectors, dichroics, beam splitters, collimators, lenses, filters, wedges, prisms, mirrors, detectors, and the like. In implementations that use an illumination system, the illumination system 1916 may be configured to direct an excitation light to reaction sites. As one example, fluorophores may be excited by green wavelengths of light; as such the wavelength of the excitation light may be approximately 1932 nm. In one implementation, the illumination system 1916 is configured to produce illumination that is parallel to a surface normal to a surface of the biosensor 1912. In another implementation, the illumination system 1916 is configured to produce illumination that is off-angle relative to the surface normal to the surface of the biosensor 1912. In yet another implementation, the illumination system 1916 is configured to produce illumination that has plural angles, including some parallel illumination and some off-angle illumination.

[00172] The system receptacle or interface 1910 is configured to engage the biosensor 1912 in at least one of a mechanical, electrical, or fluidic manner. The system receptacle 1910 may hold the biosensor 1912 in a desired orientation to facilitate the flow of fluid through the biosensor 1912. The system receptacle 1910 may also include electrical contacts that are configured to engage the biosensor 1912 so that the sequencing system 1900A may communicate with the biosensor 1912 and/or provide power to the biosensor 1912. Furthermore, the system receptacle 1910 may include fluidic ports (*e.g.*, nozzles) that are configured to engage the biosensor 1912. In some implementations, the biosensor 1912 is coupled in a removable way to the system receptacle 1910 in a mechanical manner, in an electrical manner, and also in a fluidic manner.

[00173] In addition, the sequencing system 1900A may communicate remotely with other systems or networks or with other bioassay systems 1900A. Detection data obtained by the bioassay system(s) 1900A may be stored in a remote database.

[00174] Figure 19B is a block diagram of a system controller 1906 that can be used in the system of Figure 19A. In one implementation, the system controller 1906 includes one or more processors or modules that can communicate with one another. Each of the processors or modules may include an algorithm (*e.g.*, instructions stored on a tangible and/or non-transitory computer readable storage medium) or sub-algorithms to perform particular processes. The system controller 1906 is illustrated conceptually as a collection of modules, but may be implemented utilizing any combination of dedicated hardware boards, DSPs, processors, etc. Alternatively, the system controller 1906 may be implemented utilizing an off-the-shelf PC with a single processor or multiple processors, with the functional operations distributed between the processors. As a further option, the modules described below may be implemented utilizing a hybrid configuration in which certain modular functions are performed utilizing dedicated hardware, while the remaining modular functions are performed utilizing an off-the-shelf PC and the like. The modules also may be implemented as software modules within a processing unit.

[00175] During operation, a communication port 1950 may transmit information (*e.g.*, commands) to or receive information (*e.g.*, data) from the biosensor 1912 (Figure 19A) and/or the sub-systems 1908, 1914, 1904 (Figure 19A). In implementations, the communication port 1950 may output a plurality of sequences of pixel signals. A communication link 1934 may receive user input from the user interface 1918 (Figure 19A) and transmit data or information to the user interface 1918. Data from the biosensor 1912 or sub-systems 1908, 1914, 1904 may be processed by the system controller 1906 in real-time during a bioassay session. Additionally or alternatively, data may be stored temporarily in a system memory during a bioassay session and processed in slower than real-time or off-line operation.

[00176] As shown in Figure 19B, the system controller 1906 may include a plurality of modules 1926-1948 that communicate with a main control module 1924, along with a central processing unit (CPU) 1952. The main control module 1924 may communicate with the user interface 1918 (Figure 19A). Although the modules 1926-1948 are shown as communicating directly with the main control module 1924, the modules 1926-1948 may also communicate directly with each other, the user interface 1918, and the biosensor 1912. Also, the modules 1926-1948 may communicate with the main control module 1924 through the other modules.

[00177] The plurality of modules 1926-1948 includes system modules 1928-1932, 1926 that communicate with the sub-systems 1908, 1914, 1904, and 1916. The fluidic control module 1928 may communicate with the fluidic control system 1908 to control the valves and flow sensors of

the fluid network for controlling the flow of one or more fluids through the fluid network. The fluid storage module 1930 may notify the user when fluids are low or when the waste reservoir is at or near capacity. The fluid storage module 1930 may also communicate with the temperature control module 1932 so that the fluids may be stored at a desired temperature. The illumination module 1926 may communicate with the illumination system 1916 to illuminate the reaction sites at designated times during a protocol, such as after the desired reactions (*e.g.*, binding events) have occurred. In some implementations, the illumination module 1926 may communicate with the illumination system 1916 to illuminate the reaction sites at designated angles.

[00178] The plurality of modules 1926-1948 may also include a device module 1936 that communicates with the biosensor 1912 and an identification module 1938 that determines identification information relating to the biosensor 1912. The device module 1936 may, for example, communicate with the system receptacle 1910 to confirm that the biosensor has established an electrical and fluidic connection with the sequencing system 1900A. The identification module 1938 may receive signals that identify the biosensor 1912. The identification module 1938 may use the identity of the biosensor 1912 to provide other information to the user. For example, the identification module 1938 may determine and then display a lot number, a date of manufacture, or a protocol that is recommended to be run with the biosensor 1912.

[00179] The plurality of modules 1926-1948 also includes an analysis module 1944 (also called a signal processing module or signal processor) that receives and analyzes the signal data (*e.g.*, image data) from the biosensor 1912. The analysis module 1944 includes memory (*e.g.*, RAM or Flash) to store detection/image data. Detection data can include a plurality of sequences of pixel signals, such that a sequence of pixel signals from each of the millions of sensors (or pixels) can be detected over many base calling cycles. The signal data may be stored for subsequent analysis or may be transmitted to the user interface 1918 to display desired information to the user. In some implementations, the signal data may be processed by the solid-state imager (*e.g.*, CMOS image sensor) before the analysis module 1944 receives the signal data.

[00180] The analysis module 1944 is configured to obtain image data from the light detectors at each of a plurality of sequencing cycles. The image data is derived from the emission signals detected by the light detectors. The analysis module 1944 processes the image data for each of the plurality of sequencing cycles through the neural network-based base caller 300 and produces a base call for at least some of the analytes at each of the plurality of sequencing cycles. The light detectors can be part of one or more over-head cameras (*e.g.*, Illumina's GAIIX's CCD

camera taking images of the clusters on the biosensor 1912 from the top), or can be part of the biosensor 1912 itself (*e.g.*, Illumina's iSeq's CMOS image sensors underlying the clusters on the biosensor 1912 and taking images of the clusters from the bottom).

[00181] The output of the light detectors is the sequencing images, each depicting intensity emissions of the clusters and their surrounding background. The sequencing images depict intensity emissions generated as a result of nucleotide incorporation in the sequences during the sequencing. The intensity emissions are from associated analytes and their surrounding background. The sequencing images are stored in memory 1948.

[00182] Protocol modules 1940 and 1942 communicate with the main control module 1924 to control the operation of the sub-systems 1908, 1914, and 1904 when conducting predetermined assay protocols. The protocol modules 1940 and 1942 may include sets of instructions for instructing the sequencing system 1900A to perform specific operations pursuant to predetermined protocols. As shown, the protocol module may be a sequencing-by-synthesis (SBS) module 1940 that is configured to issue various commands for performing sequencing-by-synthesis processes. In SBS, the extension of a nucleic acid primer along a nucleic acid template is monitored to determine the sequence of nucleotides in the template. The underlying chemical process can be polymerization (*e.g.*, as catalyzed by a polymerase enzyme) or ligation (*e.g.*, catalyzed by a ligase enzyme). In a particular polymerase-based SBS implementation, fluorescently labeled nucleotides are added to a primer (thereby extending the primer) in a template-dependent fashion such that the detection of the order and type of nucleotides added to the primer can be used to determine the sequence of the template. For example, to initiate a first SBS cycle, commands can be given to deliver one or more labeled nucleotides, DNA polymerase, etc., into/through a flow cell that houses an array of nucleic acid templates. The nucleic acid templates may be located at corresponding reaction sites. Those reaction sites where primer extension causes a labeled nucleotide to be incorporated can be detected through an imaging event. During an imaging event, the illumination system 1916 may provide an excitation light to the reaction sites. Optionally, the nucleotides can further include a reversible termination property that terminates further primer extension once a nucleotide has been added to a primer. For example, a nucleotide analog having a reversible terminator moiety can be added to a primer such that subsequent extension cannot occur until a deblocking agent is delivered to remove the moiety. Thus, for implementations that use reversible termination, a command can be given to deliver a deblocking reagent to the flow cell (before or after detection occurs). One or more commands can be given to effect wash(es) between the various delivery steps. The cycle can then be repeated *n* times to extend the primer by *n* nucleotides, thereby detecting a sequence of length *n*. Exemplary sequencing techniques are described, for example, in Bentley et al., Nature

4196:193-199 (20019); WO 04/0119497; US 7,019,026; WO 91/066719; WO 07/123744; US 7,329,492; US 7,211,414; US 7,311,019; US 7,401,219, and US 20019/01470190192, each of which is incorporated herein by reference.

[00183] For the nucleotide delivery step of an SBS cycle, either a single type of nucleotide can be delivered at a time, or multiple different nucleotide types (*e.g.*, A, C, T and G together) can be delivered. For a nucleotide delivery configuration where only a single type of nucleotide is present at a time, the different nucleotides need not have distinct labels since they can be distinguished based on temporal separation inherent in the individualized delivery. Accordingly, a sequencing method or apparatus can use single color detection. For example, an excitation source need only provide excitation at a single wavelength or in a single range of wavelengths. For a nucleotide delivery configuration where delivery results in multiple different nucleotides being present in the flow cell at one time, sites that incorporate different nucleotide types can be distinguished based on different fluorescent labels that are attached to respective nucleotide types in the mixture. For example, four different nucleotides can be used, each having one of four different fluorophores. In one implementation, the four different fluorophores can be distinguished using excitation in four different regions of the spectrum. For example, four different excitation radiation sources can be used. Alternatively, fewer than four different excitation sources can be used, but optical filtration of the excitation radiation from a single source can be used to produce different ranges of excitation radiation at the flow cell.

[00184] In some implementations, fewer than four different colors can be detected in a mixture having four different nucleotides. For example, pairs of nucleotides can be detected at the same wavelength but distinguished based on a difference in intensity for one member of the pair compared to the other, or based on a change to one member of the pair (*e.g.*, via chemical modification, photochemical modification or physical modification) that causes an apparent signal to appear or disappear compared to the signal detected for the other member of the pair. Exemplary apparatus and methods for distinguishing four different nucleotides using detection of fewer than four colors are described for example in US Pat. App. Ser. Nos. 61/19319,294 and 61/619,19719, which are incorporated herein by reference in their entireties. U.S. Application No. 13/624,200, which was filed on September 21, 2012, is also incorporated by reference in its entirety.

[00185] The plurality of protocol modules may also include a sample-preparation (or generation) module 1942 that is configured to issue commands to the fluidic control system 1908 and the temperature control system 1904 for amplifying a product within the biosensor 1912. For example, the biosensor 1912 may be engaged to the sequencing system 1900A. The amplification module 1942 may issue instructions to the fluidic control system 1908 to deliver

necessary the amplification components to reaction chambers within the biosensor 1912. In other implementations, the reaction sites may already contain some components for amplification, such as the template DNA and/or primers. After delivering the amplification components to the reaction chambers, the amplification module 1942 may instruct the temperature control system 1904 to cycle through different temperature stages according to known amplification protocols. In some implementations, the amplification and/or nucleotide incorporation is performed isothermally.

[00186] The SBS module 1940 may issue commands to perform bridge PCR where clusters of clonal amplicons are formed on localized areas within a channel of a flow cell. After generating the amplicons through bridge PCR, the amplicons may be “linearized” to make single stranded template DNA, or sstDNA, and a sequencing primer may be hybridized to a universal sequence that flanks a region of interest. For example, a reversible terminator-based sequencing by synthesis method can be used as set forth above or as follows.

[00187] Each base calling or sequencing cycle can extend an sstDNA by a single base which can be accomplished for example by using a modified DNA polymerase and a mixture of four types of nucleotides. The different types of nucleotides can have unique fluorescent labels, and each nucleotide can further have a reversible terminator that allows only a single-base incorporation to occur in each cycle. After a single base is added to the sstDNA, an excitation light may be incident upon the reaction sites and fluorescent emissions may be detected. After detection, the fluorescent label and the terminator may be chemically cleaved from the sstDNA. Another similar base calling or sequencing cycle may follow. In such a sequencing protocol, the SBS module 1940 may instruct the fluidic control system 1908 to direct a flow of reagent and enzyme solutions through the biosensor 1912. Exemplary reversible terminator-based SBS methods which can be utilized with the apparatus and methods set forth herein are described in US Patent Application Publication No. 2007/0166705 A1, US Patent Application Publication No. 2006/0188901 A1, U.S. Pat. No. 7,057,026, US Patent Application Publication No. 2006/0240439 A1, US Patent Application Publication No. 2006/0281109 A1, PCT Publication No. WO 05/065814, US Patent Application Publication No. 2005/0100900 A1, PCT Publication No. WO 06/064199 and PCT Publication No. WO 07/010251, each of which is incorporated herein by reference in its entirety. Exemplary reagents for reversible terminator-based SBS are described in U.S. Pat. No. 7,541,444; U.S. Pat. No. 7,057,026; U.S. Pat. No. 7,414,116; U.S. Pat. No. 7,427,673; U.S. Pat. No. 7,566,537; U.S. Pat. No. 7,592,435 and WO 07/135368, each of which is incorporated herein by reference in its entirety.

[00188] In some implementations, the amplification and SBS modules may operate in a single assay protocol where, for example, a template nucleic acid is amplified and subsequently sequenced within the same cartridge.

[00189] The sequencing system 1900A may also allow the user to reconfigure an assay protocol. For example, the sequencing system 1900A may offer options to the user through the user interface 1918 for modifying the determined protocol. For example, if it is determined that the biosensor 1912 is to be used for amplification, the sequencing system 1900A may request a temperature for the annealing cycle. Furthermore, the sequencing system 1900A may issue warnings to a user if a user has provided user inputs that are generally not acceptable for the selected assay protocol.

[00190] In implementations, the biosensor 1912 includes millions of sensors (or pixels), each of which generates a plurality of sequences of pixel signals over successive base calling cycles. The analysis module 1944 detects the plurality of sequences of pixel signals and attributes them to corresponding sensors (or pixels) in accordance with the row-wise and/or column-wise location of the sensors on an array of sensors.

[00191] Figure 19C is a simplified block diagram of a system for the analysis of sensor data from the sequencing system 1900A, such as base call sensor outputs. In the example of Figure 19C, the system includes the configurable processor 1946. The configurable processor 1946 can execute a base caller (*e.g.*, the neural network-based base caller 300) in coordination with a runtime program/logic 1980 executed by the central processing unit (CPU) 1952 (*i.e.*, a host processor). The sequencing system 1900A comprises the biosensor 1912 and flow cells. The flow cells can comprise one or more tiles in which clusters of genetic material are exposed to a sequence of analyte flows used to cause reactions in the clusters to identify the bases in the genetic material. The sensors sense the reactions for each cycle of the sequence in each tile of the flow cell to provide tile data. Genetic sequencing is a data intensive operation, which translates base call sensor data into sequences of base calls for each cluster of genetic material sensed during a base call operation.

[00192] The system in this example includes the CPU 1952, which executes a runtime program/logic 1980 to coordinate the base call operations, a memory 1948B to store sequences of arrays of tile data, base call reads produced by the base calling operation, and other information used in the base call operations. Also, in this illustration the system includes a memory 1948A to store a configuration file (or files), such as FPGA bit files, and model parameters for the neural networks used to configure and reconfigure the configurable processor 1946, and to execute the neural networks. The sequencing system 1900A can include a program

for configuring a configurable processor and in some implementations a reconfigurable processor to execute the neural networks.

[00193] The sequencing system 1900A is coupled by a bus 1989 to the configurable processor 1946. The bus 1989 can be implemented using a high throughput technology, such as, in one example, bus technology compatible with the PCIe standards (Peripheral Component Interconnect Express) currently maintained and developed by the PCI-SIG (PCI Special Interest Group). Also, in this example, a memory 1948A is coupled to the configurable processor 1946 by the bus 1993. The memory 1948A can be on-board memory, disposed on a circuit board with the configurable processor 1946. The memory 1948A is used for high-speed access by the configurable processor 1946 to working data used in the base call operation. The bus 1993 can also be implemented using a high throughput technology, such as a bus technology compatible with the PCIe standards.

[00194] Configurable processors, including field programmable gate arrays (FPGAs), coarse grained reconfigurable arrays (CGRAs), and other configurable and reconfigurable devices, can be configured to implement a variety of functions more efficiently or faster than might be achieved using a general-purpose processor executing a computer program. The configuration of configurable processors involves compiling a functional description to produce a configuration file, referred to sometimes as a bitstream or bit file, and distributing the configuration file to the configurable elements on the processor. The configuration file defines the logic functions to be executed by the configurable processor, by configuring the circuit to set data flow patterns, use of distributed memory and other on-chip memory resources, lookup table contents, operations of configurable logic blocks and configurable execution units like multiply-and-accumulate units, configurable interconnects and other elements of the configurable array. A configurable processor is reconfigurable if the configuration file may be changed in the field, by changing the loaded configuration file. For example, the configuration file may be stored in volatile SRAM elements, in non-volatile read-write memory elements, and in combinations of the same, distributed among the array of configurable elements on the configurable or reconfigurable processor. A variety of commercially available configurable processors are suitable for use in a base calling operation as described herein. Examples include Google's Tensor Processing Unit (TPU)[™], rackmount solutions like GX4 Rackmount Series[™], GX9 Rackmount Series[™], NVIDIA DGX-1[™], Microsoft' Stratix V FPGA[™], Graphcore's Intelligent Processor Unit (IPU)[™], Qualcomm's Zeroth Platform[™] with Snapdragon processors[™], NVIDIA's Volta[™], NVIDIA's DRIVE PX[™], NVIDIA's JETSON TX1/TX2 MODULE[™], Intel's Nirvana[™], Movidius VPU[™], Fujitsu DPI[™], ARM's DynamicIQ[™], IBM TrueNorth[™], Lambda GPU Server with Testa V100s[™], Xilinx Alveo[™] U200, Xilinx Alveo[™] U2190, Xilinx Alveo[™]

U280, Intel/Altera Stratix™ GX2800, Intel/Altera Stratix™ GX2800, and Intel Stratix™ GX10M. In some examples, a host CPU can be implemented on the same integrated circuit as the configurable processor.

[00195] Implementations described herein implement the neural network-based base caller 300 using the configurable processor 1946. The configuration file for the configurable processor 1946 can be implemented by specifying the logic functions to be executed using a high-level description language HDL or a register transfer level RTL language specification. The specification can be compiled using the resources designed for the selected configurable processor to generate the configuration file. The same or similar specification can be compiled for the purposes of generating a design for an application-specific integrated circuit which may not be a configurable processor.

[00196] Alternatives for the configurable processor 1946, in all implementations described herein, therefore include a configured processor comprising an application specific ASIC or special purpose integrated circuit or set of integrated circuits, or a system-on-a-chip SOC device, or a graphics processing unit (GPU) processor or a coarse-grained reconfigurable architecture (CGRA) processor, configured to execute a neural network based base call operation as described herein.

[00197] In general, configurable processors and configured processors described herein, as configured to execute runs of a neural network, are referred to herein as neural network processors.

[00198] The configurable processor 1946 is configured in this example by a configuration file loaded using a program executed by the CPU 1952, or by other sources, which configures the array of configurable elements 1991 (*e.g.*, configuration logic blocks (CLB) such as look up tables (LUTs), flip-flops, compute processing units (PMUs), and compute memory units (CMUs), configurable I/O blocks, programmable interconnects) on the configurable processor to execute the base call function. In this example, the configuration includes a data flow logic 1997 which is coupled to the buses 1989 and 1993 and executes functions for distributing data and control parameters among the elements used in the base call operation.

[00199] Also, the configurable processor 1946 is configured with a data flow logic 1997 to execute the neural network-based base caller 300. The logic 1997 comprises multi-cycle execution clusters (*e.g.*, 1979) which, in this example, include execution cluster 1 through execution cluster X. The number of multi-cycle execution clusters can be selected according to a trade-off involving the desired throughput of the operation, and the available resources on the configurable processor 1946.

[00200] The multi-cycle execution clusters are coupled to the data flow logic 1997 by data flow paths 1999 implemented using configurable interconnect and memory resources on the configurable processor 1946. Also, the multi-cycle execution clusters are coupled to the data flow logic 1997 by control paths 1995 implemented using configurable interconnect and memory resources for example on the configurable processor 1946, which provide control signals indicating available execution clusters, readiness to provide input units for the execution of a run of the neural network-based base caller 300 to the available execution clusters, readiness to provide trained parameters for the neural network-based base caller 300, readiness to provide output patches of base call classification data, and other control data used for the execution of the neural network-based base caller 300.

[00201] The configurable processor 1946 is configured to execute runs of the neural network-based base caller 300 using trained parameters to produce classification data for the sensing cycles of the base calling operation. A run of the neural network-based base caller 300 is executed to produce classification data for a subject sensing cycle of the base calling operation. A run of the neural network-based base caller 300 operates on a sequence including a number N of arrays of tile data from respective sensing cycles of N sensing cycles, where the N sensing cycles provide sensor data for different base call operations for one base position per operation in a time sequence in the examples described herein. Optionally, some of the N sensing cycles can be out of sequence if needed according to a particular neural network model being executed. The number N can be any number greater than one. In some examples described herein, sensing cycles of the N sensing cycles represent a set of sensing cycles for at least one sensing cycle preceding the subject sensing cycle and at least one sensing cycle following the subject cycle in a time sequence. Examples are described herein in which the number N is an integer equal to or greater than five.

[00202] The data flow logic 1997 is configured to move tile data and at least some trained parameters of the model parameters from the memory 1948A to the configurable processor 1946 for runs of the neural network-based base caller 300, using input units for a given run including tile data for spatially aligned patches of the N arrays. The input units can be moved by direct memory access operations in one DMA operation, or in smaller units moved during available time slots in coordination with the execution of the neural network deployed.

[00203] Tile data for a sensing cycle as described herein can comprise an array of sensor data having one or more features. For example, the sensor data can comprise two images which are analyzed to identify one of four bases at a base position in a genetic sequence of DNA, RNA, or other genetic material. The tile data can also include metadata about the images and the sensors. For example, in implementations of the base calling operation, the tile data can comprise

information about the alignment of the images with the clusters such as distance from center information indicating the distance of each pixel in the array of sensor data from the center of a cluster of genetic material on the tile.

[00204] During the execution of the neural network-based base caller 300 as described below, tile data can also include data produced during the execution of the neural network-based base caller 300, referred to as intermediate data, which can be reused rather than recomputed during a run of the neural network-based base caller 300. For example, during the execution of the neural network-based base caller 300, the data flow logic 1997 can write intermediate data to the memory 1948A in place of the sensor data for a given patch of an array of tile data.

Implementations like this are described in more detail below.

[00205] As illustrated, a system is described for the analysis of base call sensor output, comprising a memory (*e.g.*, 1948A) accessible by the runtime program/logic 1980 storing tile data including sensor data for a tile from sensing cycles of a base calling operation. Also, the system includes a neural network processor, such as the configurable processor 1946 having access to the memory. The neural network processor is configured to execute runs of a neural network using trained parameters to produce classification data for sensing cycles. As described herein, a run of the neural network is operating on a sequence of N arrays of tile data from respective sensing cycles of N sensing cycles, including a subject cycle, to produce the classification data for the subject cycle. The data flow logic 1997 is provided to move tile data and the trained parameters from the memory to the neural network processor for runs of the neural network using input units including data for spatially aligned patches of the N arrays from respective sensing cycles of N sensing cycles.

[00206] Also, a system is described in which the neural network processor has access to the memory, and includes a plurality of execution clusters, the execution clusters in the plurality of execution clusters configured to execute a neural network. The data flow logic 1997 has access to the memory and to execution clusters in the plurality of execution clusters, to provide input units of tile data to available execution clusters in the plurality of execution clusters, the input units including a number N of spatially aligned patches of arrays of tile data from respective sensing cycles, including a subject sensing cycle, and to cause the execution clusters to apply the N spatially aligned patches to the neural network to produce output patches of classification data for the spatially aligned patch of the subject sensing cycle, where N is greater than 1.

[00207] Figure 20A is a simplified diagram showing aspects of the base calling operation, including functions of a runtime program (*e.g.*, the runtime logic 1980) executed by a host processor. In this diagram, the output of image sensors from a flow cell is provided on the lines 2000 to the image processing threads 2001, which can perform processes on images such as

alignment and arrangement in an array of sensor data for the individual tiles and resampling of images. The output of image sensors can also be used by processes which calculate a tile cluster mask for each tile in the flow cell, which identifies pixels in the array of sensor data that correspond to clusters of genetic material on the corresponding tile of the flow cell. The outputs of the image processing threads 2001 are provided on the lines 2002 to a dispatch logic 2003 in the CPU which routes the arrays of tile data to a data cache 2005 (*e.g.*, SSD storage) on a high-speed bus 2004, or on high-speed bus 2006 to the neural network processor hardware 2007, such as the configurable processor 1946 of Figure 19C, according to the state of the base calling operation. The processed and transformed images can be stored on the data cache 2005 for sensing cycles that were previously used. The hardware 2007 returns classification data output by the neural network to the dispatch logic 2003, which passes the information to the data cache 2005, or on lines 2008 to threads 2009 that perform base call and quality score computations using the classification data, and can arrange the data in standard formats for base call reads. The outputs of the threads 2009 that perform base calling and quality score computations are provided on the lines 2010 to threads 2011 that aggregate the base call reads, perform other operations such as data compression, and write the resulting base call outputs to specified destinations for utilization by the customers.

[00208] In some implementations, the host can include threads (not shown) that perform final processing of the output of the hardware 2007 in support of the neural network. For example, the hardware 2007 can provide outputs of classification data from a final layer of the multi-cluster neural network. The host processor can execute an output activation function, such as a softmax function, over the classification data to configure the data for use by the base call and quality score threads 2002. Also, the host processor can execute input operations (not shown), such as batch normalization of the tile data prior to input to the hardware 2007.

[00209] Figure 20B is a simplified diagram of a configuration of a configurable processor 1946 such as that of Figure 19C. In Figure 20B, the configurable processor 1946 comprises an FPGA with a plurality of high speed PCIe interfaces. The FPGA is configured with a wrapper 2090 which comprises the data flow logic 1997 described with reference to Figure 19C. The wrapper 2090 manages the interface and coordination with a runtime program in the CPU across the CPU communication link 2077 and manages communication with the on-board DRAM 2099 (*e.g.*, a memory 1448A) via the DRAM communication link 2097. The data flow logic 1997 in the wrapper 2090 provides patch data retrieved by traversing the arrays of tile data on the on-board DRAM 2099 for the number N cycles to a cluster 2085 and retrieves process data 2087 from the cluster 2085 for delivery back to the on-board DRAM 2099. The wrapper 2090 also manages the transfer of data between the on-board DRAM 2099 and host memory, for both the

input arrays of tile data, and for the output patches of classification data. The wrapper transfers patch data on the line 2083 to the allocated cluster 2085. The wrapper provides trained parameters, such as weights and biases on the line 2081 to the cluster 2085 retrieved from the on-board DRAM 2099. The wrapper provides configuration and control data on the line 2079 to the cluster 2085 provided from, or generated in response to, the runtime program on the host via the CPU communication link 2077. The cluster can also provide status signals on the line 2089 to the wrapper 2090, which are used in cooperation with control signals from the host to manage traversal of the arrays of tile data to provide spatially aligned patch data, and to execute the multi-cycle neural network over the patch data using the resources of the cluster 2085.

[00210] As mentioned above, there can be multiple clusters on a single configurable processor managed by the wrapper 2090 configured for executing on corresponding ones of multiple patches of the tile data. Each cluster can be configured to provide classification data for base calls in a subject sensing cycle using the tile data of multiple sensing cycles described herein.

[00211] In examples of the system, model data, including kernel data like filter weights and biases, can be sent from the host CPU to the configurable processor, so that the model can be updated as a function of the cycle number. A base calling operation can comprise, for a representative example, on the order of hundreds of sensing cycles. A base calling operation can include paired end reads in some implementations. For example, the model-trained parameters may be updated once every 20 cycles (or other number of cycles), or according to update patterns implemented for particular systems and neural network models. In some implementations, including paired end reads, in which a sequence for a given string in a genetic cluster on a tile includes a first part extending from a first end down (or up) the string, and a second part extending from a second end up (or down) the string, the trained parameters can be updated on the transition from the first part to the second part.

[00212] In some examples, image data for multiple cycles of sensing data for a tile can be sent from the CPU to the wrapper 2090. The wrapper 2090 can optionally do some pre-processing and transformation of the sensing data and write the information to the on-board DRAM 2099. The input tile data for each sensing cycle can include arrays of sensor data including on the order of 4000 x 3000 pixels per sensing cycle per tile or more, with two features representing the colors of two images of the tile, and one or two bytes per feature per pixel. For an implementation in which the number N is three sensing cycles to be used in each run of the multi-cycle neural network, the array of tile data for each run of the multi-cycle neural network can consume on the order of hundreds of megabytes per tile. In some implementations of the system, the tile data also includes an array of distance-from-cluster center (DFC) data, stored once per tile, or another type of metadata about the sensor data and the tiles.

[00213] In operation, when a multi-cycle cluster is available, the wrapper allocates a patch to the cluster. The wrapper fetches a next patch of tile data in the traversal of the tile and sends it to the allocated cluster along with appropriate control and configuration information. The cluster can be configured with enough memory on the configurable processor to hold a patch of data, including patches from multiple cycles in some systems, that is being worked on in place, and a patch of data that is to be worked on when the current patch of processing is finished using a ping-pong buffer technique or raster scanning technique in various implementations.

[00214] When an allocated cluster completes its run of the neural network for the current patch and produces an output patch, it will signal the wrapper. The wrapper will read the output patch from the allocated cluster, or alternatively the allocated cluster will push the data out to the wrapper. Then the wrapper will assemble output patches for the processed tile in the DRAM 2099. When the processing of the entire tile has been completed, and the output patches of data have been transferred to the DRAM, the wrapper sends the processed output array for the tile back to the host/CPU in a specified format. In some implementations, the on-board DRAM 2099 is managed by a memory management logic in the wrapper 2090. The runtime program can control the sequencing operations to complete the analysis of all the arrays of tile data for all the cycles in the run in a continuous flow to provide real time analysis.

Joint Training

[00215] Figure 21A illustrates one implementation of jointly training the neural network-based base caller 300 on reordered index images and sequencing images. Training data 2102 for training the neural network-based base caller 300 comprises index images 2112 and sequencing images 2114. To compensate for the low-sequence diversity in the index images 2112, a reordering/shuffling logic 2122 reorders the index images 2112 into reordered index images 2132. The reordered index images 2132 include the index images 2112 rearranged/sequenced in N orders.

[00216] Then, a join training 2126 is executed that trains the neural network-based base caller 300 on both the sequencing images 2114 and the reordered index images 2132. The training is based on, for example, multiple backpropagation iterations, each of which can be in turn use as input a different ordered combination and/or permutation of the index images 2112.

[00217] Figure 21B illustrates one implementation of how the reordered index images 2132 of Figure 21A are generated. In one implementation, this includes the following steps – (i) accessing a sequence of index images 2142 generated for index sequencing cycles of a sequencing run, and selecting a plurality of sub-sequences of index images 2152 from the sequence of index images, (ii) generating respective pluralities of reordered sub-sequences 2164,

2166, and 2168 of index images for respective sub-sequences of index images 2154, 2156, and 2158 in the plurality of sub-sequences of index images 2152, (iii) accessing a sequence of target images generated for target sequencing cycles of the sequencing run, and selecting a plurality of sub-sequences of target images from the sequence of target images, and (iv) training the neural network-based base caller 300 using the respective pluralities of reordered sub-sequences of index images, and the plurality of sub-sequences of target images.

[00218] The target sequencing cycles are early target sequencing cycles of the sequencing run. The respective pluralities of reordered sub-sequences of index images compensate for low-sequence diversity in the sequence of index images. The steps further include, based on the training, generating the trained neural network-based base caller 300, and applying the trained neural network-based base caller 300 for base calling at inference.

[00219] Figures 24 and 25 show various implementations of the technology disclosed.

Computer System

[00220] Figure 22 is a computer system 2200 that can be used by the sequencing system 1900A to implement the base calling techniques disclosed herein. Computer system 2200 includes at least one central processing unit (CPU) 2272 that communicates with a number of peripheral devices via bus subsystem 2255. These peripheral devices can include a storage subsystem 2210 including, for example, memory devices and a file storage subsystem 2236, user interface input devices 2238, user interface output devices 2276, and a network interface subsystem 2274. The input and output devices allow user interaction with computer system 2200. Network interface subsystem 2274 provides an interface to outside networks, including an interface to corresponding interface devices in other computer systems.

[00221] In one implementation, the system controller 1906 is communicably linked to the storage subsystem 2210 and the user interface input devices 2238.

[00222] User interface input devices 2238 can include a keyboard; pointing devices such as a mouse, trackball, touchpad, or graphics tablet; a scanner; a touch screen incorporated into the display; audio input devices such as voice recognition systems and microphones; and other types of input devices. In general, use of the term “input device” is intended to include all possible types of devices and ways to input information into computer system 2200.

[00223] User interface output devices 2276 can include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem can include an LED display, a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), a projection device, or some other mechanism for creating a visible image. The display subsystem can also provide a non-visual display such as audio output devices. In general, use of

the term “output device” is intended to include all possible types of devices and ways to output information from computer system 2200 to the user or to another machine or computer system.

[00224] Storage subsystem 2210 stores programming and data constructs that provide the functionality of some or all of the modules and methods described herein. These software modules are generally executed by deep learning processors 2278.

[00225] Deep learning processors 2278 can be Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), and/or Coarse-Grained Reconfigurable Architectures (CGRAs). Deep learning processors 2278 can be hosted by a deep learning cloud platform such as Google Cloud Platform™, Xilinx™, and Cirrascale™. Examples of deep learning processors 2278 include Google’s Tensor Processing Unit (TPU)™, rackmount solutions like GX4 Rackmount Series™, GX22 Rackmount Series™, NVIDIA DGX-1™, Microsoft’s Stratix V FPGA™, Graphcore’s Intelligent Processor Unit (IPU)™, Qualcomm’s Zeroth Platform™ with Snapdragon processors™, NVIDIA’s Volta™, NVIDIA’s DRIVE PX™, NVIDIA’s JETSON TX1/TX2 MODULE™, Intel’s Nirvana™, Movidius VPU™, Fujitsu DPI™, ARM’s DynamicIQ™, IBM TrueNorth™, Lambda GPU Server with Testa V100s™, and others.

[00226] Memory subsystem 2222 used in the storage subsystem 2210 can include a number of memories including a main random access memory (RAM) 2232 for storage of instructions and data during program execution and a read only memory (ROM) 2234 in which fixed instructions are stored. A file storage subsystem 2236 can provide persistent storage for program and data files, and can include a hard disk drive, a floppy disk drive along with associated removable media, a CD-ROM drive, an optical drive, or removable media cartridges. The modules implementing the functionality of certain implementations can be stored by file storage subsystem 2236 in the storage subsystem 2210, or in other machines accessible by the processor.

[00227] Bus subsystem 2255 provides a mechanism for letting the various components and subsystems of computer system 2200 communicate with each other as intended. Although bus subsystem 2255 is shown schematically as a single bus, alternative implementations of the bus subsystem can use multiple busses.

[00228] Computer system 2200 itself can be of varying types including a personal computer, a portable computer, a workstation, a computer terminal, a network computer, a television, a mainframe, a server farm, a widely-distributed set of loosely networked computers, or any other data processing system or user device. Due to the ever changing nature of computers and networks, the description of computer system 2200 depicted in Figure 22 is intended only as a specific example for purposes of illustrating the preferred implementations of the present

invention. Many other configurations of computer system 2200 are possible having more or less components than the computer system depicted in Figure 22.

Clauses

[00229] The following clauses are part of this disclosure:

1. A computer-implemented method, including:
accessing a set of sequencing images,
wherein sequencing images in the set of sequencing images are arranged in a first order;
reordering the sequencing images in a reordered set of the sequencing images,
wherein the sequencing images in the reordered set of the sequencing images are arranged in
a second order, and
wherein the first order is different than the second order; and
processing the reordered set of the sequencing images and generating at least one base call
prediction.
2. The computer-implemented method of clause 1, further including training a neural
network-based base caller over a plurality of training iterations, wherein respective training
iterations in the plurality of training iterations process respective reordered sets of the sequencing
images through the neural network-based base caller to generate respective base call predictions,
and update weights of the neural network-based base caller based on comparison of the
respective base call predictions against respective base call ground truths.
3. The computer-implemented method of clause 2, wherein the sequencing images are
arranged at a center position, at one or more right flanking positions, and at one or more left
flanking positions.
4. The computer-implemented method of clause 3, wherein a particular one of the
sequencing images is kept fixed at the center position between the first order and the second
order.
5. The computer-implemented method of clause 4, wherein at least some of the sequencing
images at the right flanking positions and the left flanking positions are permuted between the
first order and the second order.

6. The computer-implemented method of clause 5, wherein at least some of the sequencing images at the right flanking positions and the left flanking positions are kept fixed between the first order and the second order.
7. The computer-implemented method of clause 6, further including training the neural network-based base caller over a first training iteration that processes the set of sequencing images through the neural network-based base caller to generate a first base call prediction, and updates weights of the neural network-based base caller based on comparison of the first base call prediction against a first base call ground truth.
8. The computer-implemented method of clause 7, further including training the neural network-based base caller over a second training iteration that processes the reordered set of sequencing images through the neural network-based base caller to generate a second base call prediction, and updates weights of the neural network-based base caller based on comparison of the second base call prediction against a second base call ground truth.
9. The computer-implemented method of clause 1, wherein respective ones of the sequencing images are generated for respective sequencing cycles of a sequencing run in the first order.
10. The computer-implemented method of clause 9, wherein respective ones of the sequencing images are respective index images generated for respective index sequencing cycles of the sequencing run in the first order.
11. The computer-implemented method of clause 1, further including:
selecting a plurality of subsets of N sequencing images from a set of M sequencing images,
wherein respective subsets in the plurality of subsets arrange the N sequencing images in
respective sequences of the N sequencing images; and wherein $M > N$.
12. The computer-implemented method of clause 11, wherein the respective sequences have different combinations of the N sequencing images.
13. The computer-implemented method of clause 12, wherein the combinations are combinations with replacement.
14. The computer-implemented method of clause 12, wherein the combinations are combinations without replacement.

15. The computer-implemented method of clause 11, wherein the respective sequences have different permutations of the N sequencing images.
16. The computer-implemented method of clause 15, wherein the permutations are permutations with repetition.
17. The computer-implemented method of clause 15, wherein the permutations are permutations without repetition.
18. The computer-implemented method of clause 11, wherein the respective sequences arrange the N sequencing images in accordance with consecutive sequencing cycles.
19. The computer-implemented method of clause 11, wherein the respective sequences arrange the N sequencing images in accordance with non-consecutive sequencing cycles.
20. The computer-implemented method of clause 11, wherein the respective sequences arrange the N sequencing images in accordance with a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.
21. The computer-implemented method of clause 11, wherein the N varies between the respective sequences.
22. The computer-implemented method of clause 11, wherein the N is constant between the respective sequences.
23. The computer-implemented method of clause 11, wherein the respective sequences have overlapping sequencing images.
24. The computer-implemented method of clause 11, wherein the respective sequences have no overlapping sequencing images.
25. The computer-implemented method of clause 11, further including training the neural network-based base caller over respective training iterations that process the respective sequences of the N sequencing images through the neural network-based base caller to generate respective base call predictions, and update weights (*e.g.*, via backpropagation) of the neural network-based base caller based on comparison of the respective base call predictions against respective base call ground truths.
26. The computer-implemented method of clause 11, wherein the M sequencing images are generated for consecutive sequencing cycles.

27. The computer-implemented method of clause 11, wherein the M sequencing images are generated for non-consecutive sequencing cycles.
28. The computer-implemented method of clause 11, wherein the M sequencing images are generated for a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.
29. A computer-implemented method, including:
accessing a plurality of sequences of N sequencing images; and
training a neural network-based base caller over a plurality of training iterations, wherein
respective training iterations in the plurality of training iterations process respective
sequences of the N sequencing images in the plurality of sequences in respective orders
through the neural network-based base caller to generate respective base call predictions, and
update weights (*e.g.*, via backpropagation) of the neural network-based base caller based on
comparison of the respective base call predictions against respective base call ground truths.
30. The computer-implemented method of clause 29, wherein the respective sequences
arrange the N sequencing images at a center position, at one or more right flanking positions, and
at one or more left flanking positions.
31. The computer-implemented method of clause 30, wherein a particular one of the N
sequencing images is kept fixed at the center position between the respective sequences.
32. The computer-implemented method of clause 31, wherein at least some of the N
sequencing images at the right flanking positions and the left flanking positions vary between the
respective sequences.
33. The computer-implemented method of clause 29, wherein the N sequencing images are
selected from M sequencing images, wherein $M > N$.
34. The computer-implemented method of clause 33, wherein the respective sequences have
different combinations of the N sequencing images.
35. The computer-implemented method of clause 34, wherein the combinations are
combinations with replacement.
36. The computer-implemented method of clause 34, wherein the combinations are
combinations without replacement.

37. The computer-implemented method of clause 33, wherein the respective sequences have different permutations of the N sequencing images.
38. The computer-implemented method of clause 37, wherein the permutations are permutations with repetition.
39. The computer-implemented method of clause 37, wherein the permutations are permutations without repetition.
40. The computer-implemented method of clause 29, wherein the respective sequences arrange the N sequencing images in accordance with consecutive sequencing cycles.
41. The computer-implemented method of clause 29, wherein the respective sequences arrange the N sequencing images in accordance with non-consecutive sequencing cycles.
42. The computer-implemented method of clause 29, wherein the respective sequences arrange the N sequencing images in accordance with a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.
43. The computer-implemented method of clause 29, wherein the N varies between the respective sequences.
44. The computer-implemented method of clause 29, wherein the N is constant between the respective sequences.
45. The computer-implemented method of clause 29, wherein the respective sequences have overlapping sequencing images.
46. The computer-implemented method of clause 29, wherein the respective sequences have no overlapping sequencing images.
47. The computer-implemented method of clause 33, wherein the M sequencing images are generated for consecutive sequencing cycles.
48. The computer-implemented method of clause 33, wherein the M sequencing images are generated for non-consecutive sequencing cycles.
49. The computer-implemented method of clause 33, wherein the M sequencing images are generated for a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.

50. The computer-implemented method of clause 29, wherein the M sequencing images are index sequencing images generated for index sequencing cycles of a sequencing run.

51. A computer-implemented method of training a neural network-based base caller that comprises a spatial logic configured to extract intensity features confined to per-cycle sequencing images in an input, and a temporal logic configured to extract intensity features spanning pan-cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic, wherein the training includes forward propagating sequencing images in the first training dataset through the spatial logic and the temporal logic and generating base call predictions, and backward propagating, through the temporal logic and the spatial logic, gradient updates determined based on comparison of the base call predictions against base call ground truths;

further training only the trained spatial logic of the neural network-based base caller on a second training dataset of sequencing images to generate a retrained spatial logic, wherein the further training includes forward propagating sequencing images in the second training dataset through the trained spatial logic and the trained temporal logic and generating further base call predictions, and backward propagating, only through the trained spatial logic, further gradient updates determined based on comparison of the further base call predictions against further base call ground truths, thereby bypassing backward propagating the further gradient updates through the trained temporal logic; and

applying the retrained spatial logic and the trained temporal logic of the neural network-based base caller on new sequencing images at inference to generate new base call predictions.

52. The computer-implemented method of clause 51, wherein the sequencing images in the first training dataset of sequencing images are read sequencing images generated for read sequencing cycles of a sequencing run.

53. The computer-implemented method of clause 51, wherein the sequencing images in the second training dataset of sequencing images are index sequencing images generated for index sequencing cycles of the sequencing run.

54. The computer-implemented method of clause 53, wherein the sequencing images in the second training dataset of sequencing images include a combination of the read sequencing images and the index sequencing images.

55. The computer-implemented method of clause 53, wherein the bypassing prevents the trained temporal logic from learning temporal dependencies between intensity features of the index sequencing images.

56. A computer-implemented method of training a neural network-based base caller that comprises a spatial logic configured to extract intensity features confined to per-cycle sequencing images in an input, and a temporal logic configured to extract intensity features spanning pan-cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic;
further training only the trained spatial logic of the neural network-based base caller on a second training dataset of sequencing images to generate a retrained spatial logic; and
configuring the neural network-based base caller with the retrained spatial logic and the trained temporal logic to execute future base calling.

57. A computer-implemented method of training a neural network-based base caller that comprises a spatial logic configured to extract intensity features confined to per-cycle sequencing images in an input, and a temporal logic configured to extract intensity features spanning pan-cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic;
further training only the trained spatial logic of the neural network-based base caller on a second training dataset of index sequencing images to generate a retrained spatial logic; and
configuring the neural network-based base caller with the retrained spatial logic and the trained temporal logic to execute future base calling of index sequencing cycles.

58. A system, comprising:

an accessing logic configured to access a set of sequencing images,
wherein sequencing images in the set of sequencing images are arranged in a first order;
a reordering logic configured to reorder the sequencing images in a reordered set of the sequencing images,
wherein the sequencing images in the reordered set of the sequencing images are arranged in a second order, and

wherein the first order is different than the second order; and
a processing logic configured to process the reordered set of the sequencing images and generate at least one base call prediction.

59. The system of clause 58, further comprising:

a normalization logic configured to normalize one or more of the sequencing images in the reordered set of the sequencing images.

60. The system of clause 58, wherein one or more of the sequencing images in the ordered and/or reordered set of the sequencing images are normalized sequencing images.

61. A computer-implemented method of training a neural network-based base caller to base call index sequences, the method including:

accessing a sequence of index images generated for index sequencing cycles of a sequencing run,
and selecting a plurality of sub-sequences of index images from the sequence of index images;

generating respective pluralities of reordered sub-sequences of index images for respective sub-sequences of index images in the plurality of sub-sequences of index images, wherein reordered sub-sequences of index images in a given plurality of reordered sub-sequences of index images generated for a given sub-sequence of index images each have a different ordering;

accessing a sequence of target images generated for target sequencing cycles of the sequencing run, and selecting a plurality of sub-sequences of target images from the sequence of target images;

training a neural network-based base caller using the respective pluralities of reordered sub-sequences of index images, and the plurality of sub-sequences of target images.

62. The computer-implemented method of clause 61, wherein the target sequencing cycles are early target sequencing cycles of the sequencing run.

63. The computer-implemented method of clause 61, wherein the respective pluralities of reordered sub-sequences of index images compensate for low-sequence diversity in the sequence of index images.

64. The computer-implemented method of clause 61, further including, based on the training, generating a trained neural network-based base caller, and applying the trained neural network-based base caller for base calling at inference.

65. A non-transitory computer readable storage medium impressed with computer program instructions to train a neural network-based base caller to base call index sequences, the instructions, when executed on a processor, implement a method comprising:

accessing a sequence of index images generated for index sequencing cycles of a sequencing run, and selecting a plurality of sub-sequences of index images from the sequence of index images;

generating respective pluralities of reordered sub-sequences of index images for respective sub-sequences of index images in the plurality of sub-sequences of index images, wherein reordered sub-sequences of index images in a given plurality of reordered sub-sequences of index images generated for a given sub-sequence of index images each have a different ordering;

accessing a sequence of target images generated for target sequencing cycles of the sequencing run, and selecting a plurality of sub-sequences of target images from the sequence of target images;

training a neural network-based base caller using the respective pluralities of reordered sub-sequences of index images, and the plurality of sub-sequences of target images.

66. The non-transitory computer readable storage medium of clause 64, wherein the target sequencing cycles are early target sequencing cycles of the sequencing run.

67. The non-transitory computer readable storage medium of clause 64, wherein the respective pluralities of reordered sub-sequences of index images compensate for low-sequence diversity in the sequence of index images.

68. The computer-implemented method of clause 64, implementing the method further comprising, based on the training, generating a trained neural network-based base caller, and applying the trained neural network-based base caller for base calling at inference.

69. A system including one or more processors coupled to memory, the memory loaded with computer instructions to train a neural network-based base caller to base call index sequences, the instructions, when executed on the processors, implement actions comprising:

accessing a sequence of index images generated for index sequencing cycles of a sequencing run, and selecting a plurality of sub-sequences of index images from the sequence of index images;

generating respective pluralities of reordered sub-sequences of index images for respective sub-sequences of index images in the plurality of sub-sequences of index images, wherein

reordered sub-sequences of index images in a given plurality of reordered sub-sequences of index images generated for a given sub-sequence of index images each have a different ordering;

accessing a sequence of target images generated for target sequencing cycles of the sequencing run, and selecting a plurality of sub-sequences of target images from the sequence of target images;

training a neural network-based base caller using the respective pluralities of reordered sub-sequences of index images, and the plurality of sub-sequences of target images.

70. The system of clause 69, wherein the target sequencing cycles are early target sequencing cycles of the sequencing run.

71. The system of clause 69, wherein the respective pluralities of reordered sub-sequences of index images compensate for low-sequence diversity in the sequence of index images.

72. The system of clause 69, further implementing actions comprising, based on the training, generating a trained neural network-based base caller, and applying the trained neural network-based base caller for base calling at inference.

73. A trained base caller, comprising:

neural network weights trained on training data comprising:

- (i) respective pluralities of reordered sub-sequences of index images generated for respective sub-sequences of index images in a plurality of sub-sequences of index images, wherein reordered sub-sequences of index images in a given plurality of reordered sub-sequences of index images generated for a given sub-sequence of index images each have a different ordering, and wherein the plurality of sub-sequences of index images are selected from a sequence of index images generated for index sequencing cycles of a sequencing run; and
- (ii) a plurality of sub-sequences of target images selected from a sequence of target images generated for target sequencing cycles of the sequencing run.

74. The trained base caller of clause 73, wherein the target sequencing cycles are early target sequencing cycles of the sequencing run.

[00230] Other implementations of the method described in this section can include a non-transitory computer readable storage medium storing instructions executable by a processor to perform any of the methods described above. Yet another implementation of the method described in this section can include a system including a memory and one or more processors

operable to execute instructions, stored in the memory, to perform any of the methods described above.

[00231] While the present invention is disclosed by reference to the preferred implementations and examples detailed above, it is to be understood that these examples are intended in an illustrative rather than in a limiting sense. It is contemplated that modifications and combinations will readily occur to those skilled in the art, which modifications and combinations will be within the spirit of the invention and the scope of the following clauses.

[00232] What is claimed is:

CLAIMS

1. A computer-implemented method, including:
accessing a set of sequencing images,
wherein sequencing images in the set of sequencing images are arranged in a first order;
reordering the sequencing images in a reordered set of the sequencing images,
wherein the sequencing images in the reordered set of the sequencing images are arranged in
a second order, and
wherein the first order is different than the second order; and
processing the reordered set of the sequencing images and generating at least one base call
prediction.
2. The computer-implemented method of claim 1, further including training a neural
network-based base caller over a plurality of training iterations, wherein respective training
iterations in the plurality of training iterations process respective reordered sets of the sequencing
images through the neural network-based base caller to generate respective base call predictions,
and update weights of the neural network-based base caller based on comparison of the
respective base call predictions against respective base call ground truths.
3. The computer-implemented method of claim 1 or 2, wherein the sequencing images are
arranged at a center position, at one or more right flanking positions, and at one or more left
flanking positions.
4. The computer-implemented method of claim 3, wherein a particular one of the
sequencing images is kept fixed at the center position between the first order and the second
order.
5. The computer-implemented method of claim 3 or 4, wherein at least some of the
sequencing images at the right flanking positions and the left flanking positions are permuted
between the first order and the second order.
6. The computer-implemented method of any of claims 3-5, wherein at least some of the
sequencing images at the right flanking positions and the left flanking positions are kept fixed
between the first order and the second order.
7. The computer-implemented method of any of claims 2-6, further including training the
neural network-based base caller over a first training iteration that processes the set of
sequencing images through the neural network-based base caller to generate a first base call

prediction, and updates weights of the neural network-based base caller based on comparison of the first base call prediction against a first base call ground truth.

8. The computer-implemented method of any of claims 2-7, further including training the neural network-based base caller over a second training iteration that processes the reordered set of sequencing images through the neural network-based base caller to generate a second base call prediction, and updates weights of the neural network-based base caller based on comparison of the second base call prediction against a second base call ground truth.

9. The computer-implemented method of any of claims 2-8, wherein respective ones of the sequencing images are generated for respective sequencing cycles of a sequencing run in the first order.

10. The computer-implemented method of claim 9, wherein respective ones of the sequencing images are respective index images generated for respective index sequencing cycles of the sequencing run in the first order.

11. The computer-implemented method of any of claims 1-10, further including:
selecting a plurality of subsets of N sequencing images from a set of M sequencing images,
wherein respective subsets in the plurality of subsets arrange the N sequencing images in
respective sequences of the N sequencing images; and wherein $M > N$.

12. The computer-implemented method of claim 11, wherein the respective sequences have different combinations of the N sequencing images.

13. The computer-implemented method of claim 12, wherein the combinations are combinations with replacement.

14. The computer-implemented method of claim 12, wherein the combinations are combinations without replacement.

15. The computer-implemented method of any of claims 11-14, wherein the respective sequences have different permutations of the N sequencing images.

16. The computer-implemented method of claim 15, wherein the permutations are permutations with repetition.

17. The computer-implemented method of claim 15, wherein the permutations are permutations without repetition.

18. The computer-implemented method of any of claims 11-17, wherein the respective sequences arrange the N sequencing images in accordance with consecutive sequencing cycles.
19. The computer-implemented method of any of claims 11-17, wherein the respective sequences arrange the N sequencing images in accordance with non-consecutive sequencing cycles.
20. The computer-implemented method of any of claims 11-19, wherein the respective sequences arrange the N sequencing images in accordance with a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.
21. The computer-implemented method of any of claims 11-20, wherein the N varies between the respective sequences.
22. The computer-implemented method of any of claims 11-20, wherein the N is constant between the respective sequences.
23. The computer-implemented method of any of claims 11-22, wherein the respective sequences have overlapping sequencing images.
24. The computer-implemented method of any of claims 11-22, wherein the respective sequences have no overlapping sequencing images.
25. The computer-implemented method of any of claims 11-24, further including training the neural network-based base caller over respective training iterations that process the respective sequences of the N sequencing images through the neural network-based base caller to generate respective base call predictions, and update weights of the neural network-based base caller based on comparison of the respective base call predictions against respective base call ground truths.
26. The computer-implemented method of any of claims 11-25, wherein the M sequencing images are generated for consecutive sequencing cycles.
27. The computer-implemented method of any of claims 11-25, wherein the M sequencing images are generated for non-consecutive sequencing cycles.
28. The computer-implemented method of any of claims 11-27, wherein the M sequencing images are generated for a combination of consecutive sequencing cycles and non-consecutive sequencing cycles.

29. A computer-implemented method, including:

accessing a plurality of sequences of N sequencing images; and
training a neural network-based base caller over a plurality of training iterations, wherein
respective training iterations in the plurality of training iterations process respective
sequences of the N sequencing images in the plurality of sequences in respective orders
through the neural network-based base caller to generate respective base call predictions, and
update weights of the neural network-based base caller based on comparison of the
respective base call predictions against respective base call ground truths.

30. A computer-implemented method of training a neural network-based base caller that
comprises spatial logic configured to extract intensity features confined to per-cycle sequencing
images in an input, and temporal logic configured to extract intensity features spanning pan-
cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first
training dataset of sequencing images to generate a trained spatial logic and a trained
temporal logic, wherein the training includes forward propagating sequencing images in the
first training dataset through the spatial logic and the temporal logic and generating base call
predictions, and backward propagating, through the temporal logic and the spatial logic,
gradient updates determined based on comparison of the base call predictions against base
call ground truths;

further training only the trained spatial logic of the neural network-based base caller on a second
training dataset of sequencing images to generate a retrained spatial logic, wherein the further
training includes forward propagating sequencing images in the second training dataset
through the trained spatial logic and the trained temporal logic and generating further base
call predictions, and backward propagating, only through the trained spatial logic, further
gradient updates determined based on comparison of the further base call predictions against
further base call ground truths, thereby bypassing backward propagating the further gradient
updates through the trained temporal logic; and

applying the retrained spatial logic and the trained temporal logic of the neural network-based
base caller on new sequencing images at inference to generate new base call predictions.

31. A computer-implemented method of training a neural network-based base caller that
comprises spatial logic configured to extract intensity features confined to per-cycle sequencing
images in an input, and temporal logic configured to extract intensity features spanning pan-
cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic;
further training only the trained spatial logic of the neural network-based base caller on a second training dataset of sequencing images to generate a retrained spatial logic; and
configuring the neural network-based base caller with the retrained spatial logic and the trained temporal logic to execute future base calling.

32. A computer-implemented method of training a neural network-based base caller that comprises spatial logic configured to extract intensity features confined to per-cycle sequencing images in an input, and temporal logic configured to extract intensity features spanning pan-cycle sequencing images in the input, including:

training the spatial logic and the temporal logic of the neural network-based base caller on a first training dataset of sequencing images to generate a trained spatial logic and a trained temporal logic;
further training only the trained spatial logic of the neural network-based base caller on a second training dataset of index sequencing images to generate a retrained spatial logic; and
configuring the neural network-based base caller with the retrained spatial logic and the trained temporal logic to execute future base calling of index sequencing cycles.

33. A system, comprising:

accessing logic configured to access a set of sequencing images,
wherein sequencing images in the set of sequencing images are arranged in a first order;
reordering logic configured to reorder the sequencing images in a reordered set of the sequencing images,
wherein the sequencing images in the reordered set of the sequencing images are arranged in a second order, and
wherein the first order is different than the second order; and
processing logic configured to process the reordered set of the sequencing images and generate at least one base call prediction.