BUS NETWORK FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLER

FIELD OF THE TECHNOLOGY DISCLOSED

[0001] 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 convolution neural networks for analyzing data.

PRIORITY APPLICATION

[0002] This PCT application claims priority to and benefit of U.S. Provisional Patent Application No. 62/979,411, titled "DATA COMPRESSION FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLING," filed 20 February 2020 (Attorney Docket No. ILLM 1029-1/IP-1964-PRV) and 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). The priority applications are hereby incorporated by reference for all purposes as if fully set forth herein.

[0003] This PCT application claims priority to and benefit of U.S. Provisional Patent Application No. 62/979,399, titled "SQUEEZING LAYER FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLING," filed 20 February 2020 (Attorney Docket No. ILLM 1030-1/IP-1982-PRV) and U.S. Patent Application No. 17/180,480, titled "SPLIT ARCHITECTURE FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLER," filed 19 February 2021 (Attorney Docket No. ILLM 1030-2/IP-1982-US). The priority applications are hereby incorporated by reference for all purposes as if fully set forth herein.

[0004] This PCT application claims priority to and benefit of U.S. Patent Application No. 17/180,513, titled "BUS NETWORK FOR ARTIFICIAL INTELLIGENCE-BASED BASE CALLER," filed 19 February 2021 (Attorney Docket No. ILLM 1031-2/IP-1965-US). The priority application is hereby incorporated by reference for all purposes as if fully set forth herein.

INCORPORATIONS

[0005] The following are incorporated by reference as if fully set forth herein:

[0006] U.S. Provisional 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);

[0007] U.S. Provisional 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);

[0008] U.S. Provisional 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);

[0009] U.S. Provisional 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);

[0010] U.S. Provisional 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);

[0011] U.S. Nonprovisional 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);

[0012] U.S. Nonprovisional 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);

[0013] U.S. Nonprovisional 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);

[0014] U.S. Nonprovisional 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); and

[0015] U.S. Nonprovisional 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).

BACKGROUND

[0016] The subject matter discussed in this section should not be assumed to be prior art merely as a result of its mention in this section. Similarly, a problem mentioned in this section or associated with the subject matter provided as background should not be assumed to have been previously recognized in the prior art. The subject matter in this section merely represents different approaches, which in and of themselves can also correspond to implementations of the claimed technology.

[0017] The rapid improvement in computation capability has made deep convolution neural networks (CNNs) a great success in recent years on many computer vision tasks with significantly improved accuracy. During the inference phase, many applications demand low latency processing of one image with strict power consumption requirement, which reduces the efficiency of graphics processing unit (GPU) and other general-purpose platform, bringing opportunities for specific acceleration hardware, *e.g.*, field programmable gate array (FPGA), by customizing the digital circuit specific for the deep learning algorithm inference. However, deploying CNNs on portable and embedded systems is still challenging due to large data volume, intensive computation, varying algorithm structures, and frequent memory accesses.

[0018] As convolution contributes most operations in CNNs, the convolution acceleration scheme significantly affects the efficiency and performance of a hardware CNN accelerator. Convolution involves multiply and accumulate (MAC) operations with four levels of loops that slide along kernel and feature maps. The first loop level computes the MAC of pixels within a kernel window. The second loop level accumulates the sum of products of the MAC across different input feature maps. After finishing the first and second loop levels, a final output pixel is obtained by adding the bias. The third loop level slides the kernel window within an input feature map. The fourth loop level generates different output feature maps.

[0019] FPGAs have gained increasing interests and popularity in particular to accelerate the inference tasks, due to their (1) high degree of reconfigurability, (2) faster development time compared to application specific integrated circuits (ASICs) to catch up with the rapid evolving of CNNs, (3) good performance, and (4) superior energy efficiency compared to GPUs. The high performance and efficiency of an FPGA can be realized by synthesizing a circuit that is customized for a specific computation to directly process billions of operations with the customized memory systems. For instance, hundreds to thousands of digital signal processing (DSP) blocks on modern FPGAs support the core convolution operation, *e.g.*, multiplication and addition, with high parallelism. Dedicated data buffers between external on-chip memory and on-chip processing engines (PEs) can be designed to realize the preferred dataflow by configuring tens of Mbyte on-chip block random access memories (BRAM) on the FPGA chip.

[0020] Efficient dataflow and hardware architecture of CNN acceleration are desired to minimize data communication while maximizing resource utilization to achieve high performance. An opportunity arises to design methodology and framework to accelerate the inference process of various CNN algorithms on acceleration hardware with high performance, efficiency, and flexibility.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] 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:

[0022] Figure 1A illustrates one implementation of the disclosed compression logic that generates compressed spatial map sets for a first iteration of base calling.

[0023] Figure 1B illustrates one implementation of processing the compressed spatial map sets through the disclosed temporal logic to generate temporal map sets.

[0024] Figure 1C illustrates one implementation of processing the temporal map sets through the disclosed output logic to generate base call classification data.

[0025] Figure 1D illustrates an example of a sequence of feature map volumes successively generated by a cascade of spatial convolution layers in response to processing per-cycle image patches for a subject sequencing cycle.

[0026] Figure 1E depicts an example that illustrates how 1x1 convolutions compress feature maps.

[0027] Figure 1F shows that the compression ratio achieved by the disclosed compression logic is a function of the number of compression filters applied by the disclosed compression logic.

[0028] Figure 1G shows an example softmax function.

[0029] Figure 1H depicts example per-cluster, per-cycle probability quadruples produced by the technology disclosed.

[0030] Figure 2A shows that, during a second iteration of base calling, spatial maps and corresponding compressed spatial maps are generated only for the non-overlapping sequencing cycle 6.

[0031] Figure 2B shows that the compressed spatial map sets generated during the first iteration of base calling are used in conjunction with a compressed spatial map set generated

during the second iteration of base calling to generate base calls for the center sequencing cycle 4.

[0032] Figure 2C shows that the output layer processes a final temporal map set generated during the second iteration of base calling and produces the base calls for the center sequencing cycle 4.

[0033] Figure 3A shows that, during a third iteration of base calling, spatial maps and corresponding compressed spatial maps are generated only for the non-overlapping sequencing cycle 7.

[0034] Figure 3B shows that the compressed spatial map sets generated during the first and second iterations of base calling are used in conjunction with a compressed spatial map set generated during the third iteration of base calling to generate base calls for the center sequencing cycle 5.

[0035] Figure 3C shows that the output layer processes a final temporal map set generated during the third iteration of base calling and produces the base calls for the center sequencing cycle 5.

[0036] Figure 4A shows a fourteenth iteration of base calling for base calling a center sequencing cycle 16.

[0037] Figure 4B shows that compressed spatial maps previously generated for sequencing cycles 1 to 29 are used to generate a final temporal map set for base calling the center sequencing cycle 16.

[0038] Figure 4C shows that the output layer processes the final temporal map set generated during the fourteenth iteration of base calling and produces base calls for the center sequencing cycle 16.

[0039] Figure 5A illustrates one implementation of filtering the compressed spatial map sets for the respective sequencing cycles 1, 2, 3, 4, and 5 using filtering logic to generate respective compressed, filtered spatial maps during the first iteration of base calling.

[0040] Figure 5B shows that the output layer processes the final filtered temporal map set generated during the first iteration of base calling and produces base calls for the center sequencing cycle 3.

[0041] Figure 6A illustrates one implementation of filtering the compressed spatial map sets for the respective sequencing cycles 2, 3, 4, 5, and 6 using the filtering logic to generate respective compressed, filtered spatial maps during the second iteration of base calling.

[0042] Figure 6B shows that the output layer processes the final filtered temporal map set generated during the second iteration of base calling and produces base calls for the center sequencing cycle 4.

[0043] Figure 7A illustrates one implementation of filtering the compressed spatial map sets for the respective sequencing cycles 3, 4, 5, 6 and 7 using the filtering logic to generate respective compressed, filtered spatial maps during the third iteration of base calling.

[0044] Figure 7B shows that the output layer processes the final filtered temporal map set generated during the third iteration of base calling and produces base calls for the center sequencing cycle 5.

[0045] Figure 8A shows one implementation of processing the sets of temporal feature maps generated during the first iteration of base calling through the compression logic to generate respective sets of compressed temporal feature maps.

[0046] Figure 8B shows that the output layer processes a final compressed temporal map set generated during the first iteration of base calling and produces base calls for the center sequencing cycle 3.

[0047] Figure 9A shows that the compressed temporal map sets generated during the first base calling iteration are used in conjunction with a compressed temporal map set generated during the second iteration of base calling to generate base calls for the center sequencing cycle 4.

[0048] Figure 9B shows that the output layer processes a final compressed temporal map set generated during the second iteration of base calling and produces base calls for the center sequencing cycle 4.

[0049] Figure 10A shows that the compressed temporal map sets generated during the first and second base calling iterations are used in conjunction with a compressed temporal map set generated during the third iteration of base calling to generate base calls for the center sequencing cycle 5.

[0050] Figure 10B shows that the output layer processes a final compressed temporal map set generated during the third iteration of base calling and produces base calls for the center sequencing cycle 5.

[0051] Figure 11A shows one implementation of processing the sets of filtered temporal feature maps generated during the first iteration of base calling through the compression logic to generate respective sets of compressed, filtered temporal feature maps.

[0052] Figure 11B shows that the output layer processes a final compressed, filtered temporal map set generated during the first iteration of base calling and produces base calls for the center sequencing cycle 3.

[0053] Figure 12A shows that the compressed, filtered temporal map sets generated during the first base calling iteration are used in conjunction with a compressed, filtered temporal map

set generated during the second iteration of base calling to generate base calls for the center sequencing cycle 4.

[0054] Figure 12B shows that the output layer processes a final compressed, filtered temporal map set generated during the second iteration of base calling and produces base calls for the center sequencing cycle 4.

[0055] Figure 13A shows that the compressed, filtered temporal map sets generated during the first and second base calling iterations are used in conjunction with a compressed, filtered temporal map set generated during the third iteration of base calling to generate base calls for the center sequencing cycle 5.

[0056] Figure 13B shows that the output layer processes a final compressed, filtered temporal map set generated during the third iteration of base calling and produces base calls for the center sequencing cycle 5.

[0057] Figure 14 illustrates a first example architecture of the neural network-based base caller disclosed herein.

[0058] Figure 15 illustrates a second example architecture of the neural network-based base caller disclosed herein.

[0059] Figure 16 illustrates a third example architecture of the neural network-based base caller disclosed herein.

[0060] Figure 17 illustrates a fourth example architecture of the neural network-based base caller disclosed herein.

[0061] Figure 18 shows one implementation of a filter configuration logic that configures a count (or numerosity) of convolution filters in the compression layer in dependence upon a number of channels in the input data.

[0062] Figures 19A and 19B depict one implementation of a sequencing system. The sequencing system comprises a configurable processor.

[0063] Figure 19C is a simplified block diagram of a system for analysis of sensor data from the sequencing system, such as base call sensor outputs.

[0064] Figure 20A is a simplified diagram showing aspects of the base calling operation, including functions of a runtime program executed by a host processor.

[0065] Figure 20B is a simplified diagram of a configuration of a configurable processor.

[0066] Figure 21 illustrates another implementation of the disclosed data flow logic making compressed spatial maps, generated during the first base calling iteration, available during the second base calling iteration from off-chip memory (*e.g.*, off-chip DRAM, host RAM, host high bandwidth memory (HBM)).

[0067] Figure 22 illustrates one implementation of a disclosed data flow logic making compressed spatial maps, generated during the first base calling iteration, available during the second base calling iteration from on-chip memory (*e.g.*, on-chip DRAM, on-chip SRAM, on-chip BRAM, DRAM attached to the processor via an interconnect).

[0068] Figure 23 illustrates one implementation of a so-called split architecture of the disclosed neural network-based base caller.

[0069] Figure 24A depicts a residual connection that reinjects prior information downstream via feature-map addition.

[0070] Figure 24B depicts one implementation of residual blocks and skip connections.

[0071] Figure 24C shows a residual architecture of the neural network-based base caller in which the spatial convolution layers are grouped into residual blocks with skip connections.

[0072] Figure 25A shows details of a disclosed bus network of the neural network-based base caller described herein.

[0073] Figure 25B shows an example operation of the disclosed bus network.

[0074] Figure 25C shows one implementation of a dimension compatibility logic of the disclosed bus network.

[0075] Figure 26 shows another example of the disclosed bus network.

[0076] Figure 27 shows yet another example of the disclosed bus network.

[0077] Figure 28 shows one implementation of a scaling logic of the disclosed bus network.

[0078] Figure 29 shows one implementation of skip connections between temporal convolution layers of the temporal network.

[0079] Figure 30 compares base calling performance by the network network-based base caller configured with the compression logic (sqz2 base caller) against the network network-based base caller without the compression logic and against Illumina's non-neural network-based base caller Real-Time Analysis (RTA) software.

[0080] Figure 31 shows savings in RAM and DRAM usage brought about by the use of the disclosed compression logic.

[0081] Figure 32 compares base calling performance by the network network-based base caller configured with the split and skip architectures (split_res) against the RTA base caller and another version of the network network-based base caller without the split and skip architectures (distilled).

[0082] Figure 33 is a computer system that can be used to implement the technology disclosed.

DETAILED DESCRIPTION

[0083] 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.

Sequencing Images

[0084] Base calling is the process of determining the nucleotide composition of a sequence. 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.

[0085] The following discussion outlines how the sequencing images are generated and what they depict, in accordance with one implementation.

[0086] 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 have a 3' removable block that anchors a fluorophore signal of the nucleotide type.

[0087] 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.

[0088] 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 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.

[0089] 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.

[0090] 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.

[0091] 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 x h of the grid of pixelated units, where w (width) and h (height) are any numbers ranging from 1 and 100,000 (*e.g.*, 115 x 115, 200 x 200, 1800 x 2000, 2200 x 25000, 2800 x 3600, 4000 x 400). In some implementations, w and h are the same. In other implementations, w and h are different. The sequencing images depict intensity emissions generated as a result of nucleotide incorporation in the nucleotide sequences during the sequencing run. The intensity emissions are from associated clusters and their surrounding background.

Neural Network-Based Base Calling

[0092] The following discussion focuses on a neural network-based base caller 100 described herein. First, the input to the neural network-based base caller 100 is described, in accordance with one implementation. Then, examples of the structure and form of the neural network-based base caller 100 are provided. Finally, the output of the neural network-based base caller 100 is described, in accordance with one implementation.

[0093] A data flow logic provides the sequencing images to the neural network-based base caller 100 for base calling. The neural network-based base caller 100 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 x r of the sub-grid of pixelated units, where q (width) and r (height) are any numbers ranging from 1 and 10000 (*e.g.*, 3 x 3, 5 x 5, 7 x 7, 10 x 10, 15 x 15, 25 x 25, 64 x 64, 78 x 78, 115 x 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).

[0094] Sequencing produces m sequencing images per sequencing cycle for 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 at 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.

[0095] 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 100.

[0096] The input image data to the neural network-based base caller 100 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.

[0097] 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.

[0098] 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), and (iv) data for a second right flanking/context/next/successive/subsequent (time *t*+2) sequencing cycle.

[0099] 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), (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.

[00100] The neural network-based base caller 100 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) for generating 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.

[0101] In one implementation, the neural network-based base caller 100 outputs a base call for a single target cluster for a particular sequencing cycle. In another implementation, the neural network-based base caller 100 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 100 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.

[0102] In one implementation, the neural network-based base caller 100 is a multilayer perceptron (MLP). In another implementation, the neural network-based base caller 100 is a feedforward neural network. In yet another implementation, the neural network-based base caller 100 is a fully-connected neural network. In a further implementation, the neural network-based base caller 100 is a fully convolution neural network. In yet further implementation, the neural network-based base caller 100 is a semantic segmentation neural network. In yet another further implementation, the neural network-based base caller 100 is a generative adversarial network (GAN).

[0103] In one implementation, the neural network-based base caller 100 is a convolution neural network (CNN) with a plurality of convolution layers. In another implementation, the neural network-based base caller 100 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 100 includes both a CNN and an RNN.

[0104] In yet other implementations, the neural network-based base caller 100 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 100 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 100 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 100 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.

[0105] The neural network-based base caller 100 is trained using backpropagation-based gradient update techniques. Example gradient descent techniques that can be used for training the

neural network-based base caller 100 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 100 are Momentum, Nesterov accelerated gradient, Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam, and AMSGrad.

[0106] In one implementation, the neural network-based base caller 100 uses a specialized architecture to segregate 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 100 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 provides sequence-specific context. The neural network-based base caller 100 learns the sequence-specific context during training and base calls them. Furthermore, data for pre and post sequencing cycles provides second order contribution of pre-phasing and phasing signals to the current sequencing cycle.

[0107] 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.

[0108] 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.

[0109] Consider, for example, that the input image data comprises (i) current image patch for a current (time t) sequencing cycle to be base called, (ii) previous image patch for a previous (time t-1) sequencing cycle, and (iii) 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.

[0110] 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.

[0111] The neural network-based base caller 100 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.

[0112] 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.

[0113] 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.

[0114] 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.

[0115] Additional details about the neural network-based base caller 100 can 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.

Compression Network

[0116] As discussed above, the specialized architecture of the neural network-based base caller 100 processes sliding windows of image patches for corresponding sequencing cycles. Overlap exists between sequencing cycles of successive sliding windows. This causes the neural network-based base caller 100 to redundantly process image patches for the overlapping sequencing cycles. This in turn results in waste of compute resources. For example, in one implementation, each spatial convolution layer of the neural network-based base caller 100 has nearly 100 million multiplication operations. Then, for a window of five sequencing cycles and a cascade (or sequence) of seven spatial convolution layers, the spatial convolution neural network executes about 620 million multiplication operations. Furthermore, the temporal convolution neural network executes about 10 million multiplication operations.

[0117] Since the image data for cycle N-1 in a current sliding window (or a current iteration of base calling) is processed as cycle N in the previous sliding window (or a previous iteration of base calling), an opportunity arises to store the intermediate results of the processing done in the current sliding window and the intermediate results them in subsequent sliding windows, and thereby bypass (or obviate) redundant processing (or reprocessing) of input image data for overlapping sequencing cycles between successive sliding windows.

[0118] However, the intermediate results are several terabytes of data that require impractical amount of storage. To overcome this technical problem, the technology disclosed proposes compressing the intermediate results the first time the intermediate results are generated by the neural network-based base caller 100 and repurposing the compressed intermediate results in subsequent sliding windows to avoid redundant computation, and thereby not regenerating (or only-once generating) the intermediate results. In some implementations, the technology disclosed saves about 80% of convolutions in the spatial network of the neural network-based base caller 100. In one implementation, the 80% savings are observed in the spatial convolutions when the compression logic and repurposing of the compressed feature maps in subsequent sequencing cycles is used for an input window of five sequencing cycles (*e.g.*, cycle N, cycle N+1, cycle N-1, cycle N+2, and cycle N-2). In another implementation, 90% savings are observed in the spatial convolutions when the compression logic and repurposing of the compressed feature maps in subsequent sequencing cycles is used for an input window of ten sequencing cycles (*e.g.*, cycle N, cycle N+1, cycle N-1, cycle N+2, cycle N-2, cycle N+3, and cycle N-3). That is, the larger the window size, the bigger the savings from the use of the compression logic and repurposing of the compressed feature maps, and the larger the window size, the better the base calling performance due to incorporation of greater context from

additional flanking cycles. So bigger savings for bigger windows improves overall performance for a given compute capability.

[0119] The compute efficiency and compact compute footprint brought about by the compression logic facilitates hardware implementation of the neural network-based base caller 100 on resource-constrained processors like Central Processing Units (CPUs), Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), Coarse-Grained Reconfigurable Architectures (CGRAs), Application-Specific Integrated Circuits (ASICs), Application Specific Instruction-set Processor (ASIP), and Digital Signal Processors (DSPs).

[0120] The compute saved by the compression logic allows for incorporating more convolution operators in the neural network-based base caller 100. Examples include adding more convolution filters in the spatial and temporal convolution layers, increasing the size of the convolution filters, and increasing the number of spatial and temporal convolution layers. Additional convolution operations improve intensity pattern detection and overall base calling accuracy of the neural network-based base caller 100.

[0121] The compute saved by the compression logic also allows for expanding the input image data for a subject sliding window to include increased number of sequencing cycles. Expanded sliding windows broaden the base calling context by bringing in surplus image patches from additional flanking sequencing cycles.

[0122] Furthermore, any dip in accuracy that may occur due to the use of compressed intermediate results, as opposed to the original intermediate results, is compensated by the incorporation of additional convolution operators and expansion of the sliding windows.

[0123] Figure 1A illustrates one implementation of the disclosed compression logic that generates compressed spatial map sets for a first iteration of base calling. In the illustrated example, a first window of sequencing cycles includes sequencing cycles 1, 2, 3, 4, and 5. Respective images patches 102, 112, 122, 132, and 142 (or per-cycle analyte channel sets) for the respective sequencing cycles 1, 2, 3, 4, and 5 are separately processed through a spatial logic 104 (or spatial network or spatial subnetwork or spatial convolution neural network) to generate respective spatial maps 106, 116, 126, 136, and 146 (or intermediate results or spatial output sets or spatial feature map sets) for the respective sequencing cycles 1, 2, 3, 4, and 5. The spatial convolution network 104 can use 1D, 2D, or 3D convolutions.

[0124] The spatial logic 104 includes a sequence (or cascade) of spatial convolution layers. Each spatial convolution layer has a filter bank with a plurality of spatial convolution filters that implement segregated convolutions. Accordingly, each spatial convolution layer produces as output a plurality of spatial feature maps. The number of spatial feature maps produced by a subject spatial convolution layer is a function of the number of spatial convolution filters

configured in the subject spatial convolution layer. For example, if the subject spatial convolution layer has fourteen spatial convolution filters, then the subject spatial convolution layer produces fourteen spatial feature maps. From an aggregate perspective, the fourteen spatial feature maps can be considered a spatial feature map volume (or tensor) with fourteen channels (or depth dimension = fourteen).

[0125] Furthermore, a next spatial convolution layer that follows the subject spatial convolution layer can also be configured with fourteen spatial convolution filters. In such as case, the next spatial convolution layer processes, as input, the fourteen spatial feature maps generated the subject spatial convolution layer, and itself generates fourteen new spatial feature maps as output. Figure 1A shows the five spatial feature map sets 106, 116, 126, 136, and 146 generated by a final spatial convolution layer of the spatial network 104 for the respective sequencing cycles 1, 2, 3, 4, and 5. In the illustrated example, each of the five spatial feature map sets 106, 116, 126, 136, and 146 have fourteen feature maps.

[0126] Figure 1D illustrates a sequence of seven spatial feature map sets 196a, 196b, 196c, 196d, 196e, 196f, and 196g generated by a cascade of seven spatial convolution layers of the spatial network 104. A per-cycle input patch data 194 for a subject sequencing cycle *i* has a spatial dimensionality of 115 x 115 and a depth dimensionality of two (due to the two image channels in the original sequencing images). In one implementation, each of the seven spatial convolution layers uses 3 x 3 convolutions that reduce the spatial dimensionality of successive spatial feature map volumes by two, for example, from 10 x 10 to 8 x 8.

[0127] The first spatial feature map volume 196a has spatial dimensions 113 x 113 (*i.e.*, reduced from 115 x 115 by the 3 x 3 convolutions of the first spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the first spatial convolution layer). The second spatial feature map volume 196b has spatial dimensions 111 x 111 (*i.e.*, reduced from 113 x 113 by the 3 x 3 convolutions of the second spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the second spatial convolution layer). The third spatial feature map volume 196c has spatial dimensions 109 x 109 (*i.e.*, reduced from 111 x 111 by the 3 x 3 convolutions of the third spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the third spatial convolution layer). The fourth spatial feature map volume 196d has spatial dimensions 107 x 107 (*i.e.*, reduced from 109 x 109 by the 3 x 3 convolutions of the fourth spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the fourth spatial convolution layer). The fifth spatial feature map volume 196e has spatial dimensions 105 x 105 (*i.e.*, reduced

from 107 x 107 by the 3 x 3 convolutions of the fifth spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the fifth spatial convolution layer). The sixth spatial feature map volume 196f has spatial dimensions 103 x 103 (*i.e.*, reduced from 105 x 105 by the 3 x 3 convolutions of the sixth spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the sixth spatial convolution layer). The seventh spatial feature map volume 196g has spatial dimensions 101 x 101 (*i.e.*, reduced from 103 x 103 by the 3 x 3 convolutions of the seventh spatial convolution layer) and a depth dimension of 14 (*i.e.*, fourteen feature maps or fourteen channels due to fourteen spatial convolution filters in the seventh spatial convolution layer).

[0128] Analogizing to the multi-cycle example illustrated in Figure 1A, for the five sequencing cycles 1, 2, 3, 4, and 5 and the five per-cycle images patches 102, 112, 122, 132, and 142, the spatial logic 104 separately produces five respective sequences of the seven spatial feature map volumes 196a, 196b, 196c, 196d, 196e, 196f, and 196g, with the spatial maps 106, 116, 126, 136, and 146 in Figure 1A being equivalent to five separate instances of the final spatial feature map volume 196g in Figure 1D.

[0129] A compression logic 108 (or compression network or compression subnetwork or compression layer or squeezing layer) processes the outputs of the spatial logic 104 and generates a compressed representation of the outputs. In one implementation, the compression network 108 comprises a compression convolution layer that reduces the depth dimensionality of feature maps generated by the spatial network 104.

[0130] For example, in Figure 1A, the depth dimensionality of the spatial maps 106, 116, 126, 136, and 146 is 14 (*i.e.*, fourteen feature maps or fourteen channels per spatial output). The compression network 108 attenuates the spatial maps 106, 116, 126, 136, and 146 into respective compressed spatial map sets 110, 120, 130, 140, and 150 for the respective sequencing cycles 1, 2, 3, 4, and 5. Each of the compressed spatial map sets 110, 120, 130, 140, and 150 has a depth dimensionality of 2 (*i.e.*, two feature maps or two channels per compressed spatial output). In other implementations, the compressed spatial map sets 110, 120, 130, 140, and 150 can have a depth dimensionality of 3 or 4 (*i.e.*, three or fourth feature maps or three or fourth channels per compressed spatial output). In yet other implementations, the compressed spatial map sets 110, 120, 130, 140, and 150 can have a depth dimensionality of 1 (*i.e.*, one feature map or one channel per compressed spatial output). In one implementation, the compression layer 108 does not include an activation function like ReLU. In other implementations, it can include an activation function. In other implementations, the compression logic 108 can configure the corresponding compressed spatial map sets to each have more than four feature maps.

[0131] The discussion now turns to how the compression logic 108 generates the compressed outputs.

[0132] In one implementation, the compression logic 108 uses 1x1 convolutions to reduce the number of feature maps (*i.e.*, the depth dimension or the number of channels) while introducing non-linearity. The 1x1 convolutions have a kernel size of 1. The 1x1 convolutions can transform a volume depth into another squeezed or expanded representation without changing the spatial dimensions. A 1x1 convolution operates like a fully connected linear layer across the input channels. This is useful in mapping from feature maps with many channels to fewer feature maps. In Figure 1E, a single 1x1 convolution is applied to an input tensor with two feature maps. The 1x1 convolution compresses the two-channel input to a single-channel output.

[0133] The number of compressed outputs (or compressed feature maps or compressed spatial maps or compressed temporal maps) generated by the compression layer 108 is a function of the number of 1x1 convolution filters (or compression convolution filters or compression filters) configured in the compression layer 108. In Figure 1F, the compression layer 108 has two 1x1 convolution filters 198a and 198b. The first 1x1 convolution filter 198a processes the spatial feature volume 196g with the fourteen feature maps and generates a first feature map 199a while preserving the spatial dimensionality of 101 x 101. The second 1x1 convolution filter 198b also processes the spatial feature volume 196g with the fourteen feature maps and generates a second feature map 199b while preserving the spatial dimensionality of 101 x 101. Accordingly, the compression layer 108 reduces the spatial feature volume 196g with the fourteen feature maps into a compressed output with two spatial feature maps 199a and 199b (*i.e.*, compression ratio = 7).

[0134] From the timeseries perspective, the sequencing cycle 5 is the center sequencing cycle (N), the sequencing cycles 1 and 2 are the left flanking sequencing cycles (N-1, N-2), and the sequencing cycles 4 and 5 are the left flanking sequencing cycles (N+1, N+2). Accordingly, the center compressed output 130 is generated for the center sequencing cycle (N), the left flanking compressed output 120 is generated for the left flanking sequencing cycle (N-1), the further left flanking compressed output 110 is generated for the further left flanking sequencing cycle (N-2), the right flanking compressed output 140 is generated for the right flanking sequencing cycle (N+1), and the further right flanking compressed output 150 is generated for the further right flanking sequencing cycle (N+2).

[0135] From the pipeline perspective, the neural network-based base caller 100 executes five parallel and independent pipelines that respectively process the images patches 102, 112, 122, 132, and 142 through the spatial logic 104 and the compression logic 108 (*e.g.*, as a multi-threaded execution or a multi-clustered execution based on data parallelism). Accordingly, five

compressed outputs 110, 120, 130, 140, and 150 are separately, simultaneously, and independently generated by the neural network-based base caller 100.

[0136] In some implementations, the compression layer 108 can be considered a final spatial convolution layer of the spatial network 104. In other implementations, the compression network 108 can be considered a separate network inside or outside the specialized architecture of neural network-based base caller 100.

[0137] Figure 1B illustrates one implementation of processing the compressed spatial map sets 110, 120, 130, 140, and 150 through a temporal logic 160 (or temporal network or temporal subnetwork or temporal convolution neural network) of the neural network-based base caller 100. The temporal logic 160 processes groups of successive compressed spatial map sets on a sliding window-basis. For example, in Figure 1B, the temporal logic 160 processes a first group/window of compressed spatial map sets 110, 120, and 130 for respective sequencing cycles 1, 2, and 3, and generates temporal maps 172 (or temporal map sets or temporal feature maps or temporal feature map sets) as output. The temporal logic 160 processes a second group/window of compressed spatial map sets 120, 130, and 140 for respective sequencing cycles 2, 3, and 4, and generates temporal maps 174 as output. The temporal logic 160 processes a third group/window of compressed spatial map sets 130, 140, and 150 for respective sequencing cycles 3, 4, and 5, and generates temporal maps 176 as output. The temporal convolution network 160 can use 1D, 2D, or 3D convolutions.

[0138] The three instances of the temporal logic 160 shown in Figure 1B represent three filter banks of a first temporal convolution layer of the temporal network 160. The first filter bank applies a first set of temporal convolution filters on the first group of compressed spatial maps 110, 120, and 130 and generates the first set of temporal maps 172. The second filter bank applies a second set of temporal convolution filters on the second group of compressed spatial maps 120, 130, and 140 and generates the second set of temporal maps 174. The third filter bank applies a third set of temporal convolution filters on the third group of compressed spatial maps 130, 140, and 150 and generates the third set of temporal maps 176.

[0139] The first, second, and third sets of temporal maps 172, 174, and 176 are processed as a group by the temporal logic 160 to generate temporal maps 182. The fourth instance of the temporal logic 160 shown in Figure 1B represents a second temporal convolution layer of the temporal network 160 that produces an output for all the sequencing cycles 1, 2, 3, 4, and 5 for which per-cycle image patch pairs were fed as input to the neural network-based base caller 100 in Figure 1A.

[0140] The temporal network 160 has a cascade of temporal convolution layers (*e.g.*, 2, 3, 4, 5, or more temporal convolution layers arranged in a sequence). The cascade of temporal

convolution layers process data in a hierarchical form with different levels of grouping. That is, at a given level, a sliding window approach group-wise processes inputs at the given level to generate outputs that are subsequently group-wise processed at a next level in the sliding window fashion.

[0141] The temporal convolution layers are configured with temporal convolution filters that implement combinatory convolutions. The combinatory convolutions mix information between feature maps spanning multiple sequencing cycles. The combinatory convolutions combine data between successive sequencing cycles in a subject group/window at a current level in the temporal network 160. For example, the first temporal convolution layer combines the first group of compressed spatial maps 110, 120, and 130 for the first group of sequencing cycles 1, 2, and 3 to generate the first set of temporal maps 172; combines the second group of compressed spatial maps 120, 130, and 140 for the second group of sequencing cycles 2, 3, and 4 to generate the second set of temporal maps 174; and combines the third group of compressed spatial maps 130, 140, and 150 for the third group of sequencing cycles 3, 4, and 5 to generate the third set of temporal maps 176.

[0142] The combinatory convolutions also combine data between successive groups of sequencing cycles in a subject group/window at a current level in the temporal network 160. For example, the second temporal convolution layer combines the first, second, and third sets of temporal maps 172, 174, and 176 into the final set of temporal maps 182. At level two, the first, second, and third groups/windows of sequencing cycles from level one are grouped in a first group/window of sequencing cycles 1, 2, 3, 4, and 5.

[0143] The combinatory convolutions are configured with as many kernels as the number of inputs to be combined (*i.e.*, the depth column or fibre of the temporal convolution filters is matched with the number of inputs in the subject group/window at the current level). For example, when a temporal convolution layer combines three compressed spatial maps, it uses temporal convolution filters that each have three kernels that perform element-wise multiplication and summation throughout the depth of the three compressed spatial maps.

[0144] The final set of temporal maps 182 are produced by a final (or last) temporal convolution layer of the temporal network 160. Figure 1C illustrates one implementation of processing the final temporal map sets 182 through a disclosed output logic 190 (or output layer or output network or output subnetwork) to generate base call classification data. In one implementation, a plurality of clusters is base called simultaneously for one or more sequencing cycles. In the example illustrated in Figure 1C, base calls 192 are generated for many clusters only for the center sequencing cycle 3. In other implementations, the technology disclosed causes the output logic 190 to generate, for a given window of input, base calls not only for the

center sequencing cycle but also for the flanking sequencing cycles (as indicated by optional dotted lines), in accordance with one implementation. For example, in one implementation, the technology disclosed simultaneously generates base calls for cycle N, cycle N+1, cycle N-1, cycle N+2, cycle N-2, and so on for a given input window. That is, a single forward propagation/traversal/base calling iteration of the neural network-based base caller 102 generates base calls for multiple sequencing cycles in the input window of sequencing cycles, which is referred to herein as "many-to-many base calling."

[0145] Examples of the output layer 190 include a softmax function, a log-softmax function, an ensemble output average function, a multi-layer perceptron uncertainty function, a Bayes Gaussian distribution function, and a cluster intensity function. In one implementation, the output layer 190 produces a per-cluster, per-cycle probability quadruple for each cluster and for each sequencing cycle.

[0146] The following discussion focuses on the per-cluster, per-cycle probability quadruples using the softmax function as an example. We first explain the softmax function and then the per-cluster, per-cycle probability quadruples.

[0147] Softmax function is a preferred function for multi-class classification. The softmax function calculates the probabilities of each target class over all possible target classes. The output range of the softmax function is between zero and one and the sum of all the probabilities is equal to one. The softmax function computes the exponential of the given input value and the sum of exponential values of all the input values. The ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function, referred to herein as "exponential normalization."

[0148] Formally, training a so-called softmax classifier is regression to a class probability, rather than a true classifier as it does not return the class but rather a confidence prediction of each class's probability. The softmax function takes a class of values and converts them to probabilities that sum to one. The softmax function squashes a n-dimensional vector of arbitrary real values to n-dimensional vector of real values within the range zero to one. Thus, using the softmax function ensures that the output is a valid, exponentially normalized probability mass function (nonnegative and summing to one).

[0149] Intuitively, the softmax function is a "soft" version of the maximum function. The term "soft" derives from the fact that the softmax function is continuous and differentiable. Instead of selecting one maximal element, it breaks the vector into parts of a whole with the maximal input element getting a proportionally larger value, and the other getting a less proportion of the value. The property of outputting a probability distribution makes the softmax function suitable for probabilistic interpretation in classification tasks.

[0150] Let us consider z as a vector of inputs to the softmax layer. The softmax layer units are the number of nodes in the softmax layer and therefore, the length of the z vector is the number of units in the softmax layer (if we have ten output units, then there are ten z elements).

[0151] For an n-dimensional vector $Z = [z_1, z_2, ...z_n]$, the softmax function uses exponential normalization (exp) to produce another n-dimensional vector $p(Z)$ with normalized values in the range [0, 1] and that add to unity:

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \; and, \; p(Z) \; \rightarrow \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

$$p_j = \frac{\exp^{z_j}}{\sum_{k=1}^{n} \exp^{z_k}} \forall \; j \in 1, 2, ..., n$$

[0152] Figure 1G shows an example softmax function. Softmax function is applied to three classes as $z \mapsto \text{softmax}([z; \frac{z}{10}; -2z])$. Note that the three outputs always sum to one. They thus define a discrete probability mass function.

[0153] A particular per-cluster, per-cycle probability quadruple identifies probabilities of a base incorporated in a particular cluster at a particular sequencing cycle being A, C, T, and G. When the output layer of the neural network-based base caller 100 uses a softmax function, the probabilities in the per-cluster, per-cycle probability quadruple are exponentially normalized classification scores that sum to unity. Figure 1H depicts example per-cluster, per-cycle probability quadruples 123 produced by the softmax function for cluster 1 (121, shown in brown color) and for sequencing cycles 1 through S (122), respectively. In other words, the first subset of sequencing cycles includes S sequencing cycles.

[0154] The unreliable cluster identifier 125 identifies unreliable clusters based on generating filter values from the per-cluster, per-cycle probability quadruple. In this application, the per-cluster, per-cycle probability quadruples are also referred to as base call classification scores or normalized base call classification scores or initial base call classification scores or normalized initial base call classification scores or initial base calls.

[0155] A filter calculator 127 determines a filter value for each per-cluster, per-cycle probability quadruple based on the probabilities it identifies, thereby generating a sequence of filter values for each cluster. The sequence of filter values is stored as filter values 124.

[0156] The filter value for a per-cluster, per-cycle probability quadruple is determined based on a calculation involving one or more of the probabilities. In one implementation, the

calculation used by the filter calculator 127 is subtraction. For example, in the implementation illustrated in Figure 1H, the filter value for the per-cluster, per-cycle probability quadruple is determined by subtracting a second highest one of the probabilities (shown in blue color) from a highest one of the probabilities (shown in magenta color).

[0157] In another implementation, the calculation used by the filter calculator 116 is division. For example, the filter value for the per-cluster, per-cycle probability quadruple is determined as a ratio of the highest one of the probabilities (shown in magenta color) to the second highest one of the probabilities (shown in blue color). In yet another implementation, the calculation used by the filter calculator 127 is addition. In yet further implementation, the calculation used by the filter calculator 127 is multiplication.

[0158] In one implementation, the filter calculator 127 generates the filter values 124 using a filtering function. In one example, the filtering function is a chastity filter that defines chastity as a ratio of a brightest base intensity divided by a sum of the brightest base intensity and a second brightest base intensity. In another example, the filtering function is at least one of a maximum log probability function, a minimum squared error function, average signal-to-noise ratio (SNR), and a minimum absolute error function.

[0159] The unreliable cluster identifier 125 uses the filter values 124 to identify some clusters in the plurality of clusters as unreliable clusters 128. Data identifying the unreliable clusters 128 can be in computer readable format or medium. The unreliable clusters can be identified by instrument ID, the run number on the instrument, the flow cell ID, the lane number, the tile number, the X coordinate of the cluster, the Y coordinate of the cluster, and unique molecular identifiers (UMIs). The unreliable cluster identifier 125 identifies those clusters in the plurality of clusters as unreliable clusters whose sequences of filter values contain "G" number of filter values below a threshold "H." In one implementation, the "G" ranges from 1 to 5. In another implementation, the "H" ranges from 0.5 to 0.99. In one implementation, the unreliable clusters 128 identify those pixels that correspond to (*i.e.*, depict intensity emissions of) the unreliable clusters. Such pixels are filtered out by a filtering logic 502, as describe later in this application.

[0160] Unreliable clusters are low-quality clusters that emit an amount of desired signal which is insignificant compared to background signal. The signal to noise ratio for unreliable clusters is substantially low, for example, less than 1. In some implementations, unreliable clusters may not produce any amount of a desired signal. In other implementations, unreliable clusters may produce a very low amount of signal relative to background. In one implementation, the signal is an optical signal and is intended to include, for example, fluorescent, luminescent, scatter, or absorption signals. Signal level refers to an amount or

quantity of detected energy or coded information that has a desired or predefined characteristic. For example, an optical signal can be quantified by one or more of intensity, wavelength, energy, frequency, power luminance or the like. Other signals can be quantified according to characteristics such as voltage, current, electric field strength, magnetic field strength, frequency, power, temperature, etc. Absence of signal in unreliable clusters is understood to be a signal level of zero or a signal level that is not meaningfully distinguished from noise.

[0161] There are many potential reasons for poor quality signals of unreliable clusters. If there has been a polymerase chain reaction (PCR) error in colony amplification such that a sizable proportion of the ~1000 molecules in an unreliable cluster contains a different base at a certain position, then one may observe a signal for two bases—this is interpreted as a sign of poor quality and referred to as phase error. Phase error occurs when individual molecules in an unreliable cluster do not incorporate a nucleotide in some cycle (*e.g.*, because of incomplete remove of the 3' terminators, termed phasing) and then lag behind the other molecules, or when an individual molecule incorporates more than one nucleotide in a single cycle (*e.g.*, because of incorporation of nucleotides without effective 3'-blocking, termed prephasing). This results in the loss of synchrony in the readout of the sequence copies. The proportion of sequences in unreliable clusters that are affected by phasing and pre-phasing increases with cycle number, which is a major reason why the quality of reads tends to decline at high cycle numbers.

[0162] Unreliable clusters also result from fading. Fading is an exponential decay in signal intensity of unreliable clusters as a function of cycle number. As the sequencing run progress, the strands in unreliable clusters are washed excessively, exposed to laser emissions that create reactive species, and subject to harsh environmental conditions. All of these lead to a gradual loss of fragments in unreliable clusters, decreasing their signal intensity.

[0163] Unreliable clusters also result from underdeveloped colonies, *i.e.*, small cluster sizes of unreliable clusters that produce empty or partially filled wells on a patterned flow cell. That is, in some implementations, the unreliable clusters are indicative of empty, polyclonal, and dim wells on the patterned flow cell. Unreliable clusters also result from overlapping colonies caused by unexclusive amplification. Unreliable clusters also result from under-illumination or uneven-illumination, for example, due to being located on the edges of a flow cell. Unreliable clusters also result from impurities on the flow cell that obfuscate emitted signal. Unreliable clusters also include polyclonal clusters when multiple clusters are deposited in the same well.

[0164] The first window of sequencing cycles includes sequencing cycles 1, 2, 3, 4, and 5, and the first iteration of base calling produces base calls 192 for the center sequencing cycle 3. The second window of sequencing cycles includes sequencing cycles 2, 3, 4, 5, and 6 and a second iteration of base calling produces base calls 292 for the center sequencing cycle 4.

Accordingly, sequencing cycles 2, 3, 4, and 5 are the overlapping sequencing cycles between the first and second windows or between the second and third iterations of base calling.

[0165] The disclosed base calling systems and techniques store, in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM), the compressed spatial map sets 120, 130, 140, and 150 generated during the first iteration of base calling for the respective sequencing cycles 2, 3, 4, and 5. During the second iteration of base calling, the disclosed base calling systems and techniques do not reprocess the respective input image patches 112, 122, 132, and 142 for the overlapping cycles 2, 3, 4, and 5 through the spatial network 104. Instead, during the second iteration of base calling, the disclosed base calling systems and techniques reuse the previously generated compressed spatial map sets 120, 130, 140, and 150 in lieu of the respective input image patches 112, 122, 132, and 142.

[0166] The compression logic is further configured to require the compressed spatial map sets 120, 130, 140, and 150 and the respective input image patches 112, 122, 132, and 142 to have the same number of per-cycle feature maps/channels. This ensures that the compressed spatial map sets 120, 130, 140, and 150 are lossless representatives of the respective input image patches 112, 122, 132, and 142. That is, if the respective input image patches 112, 122, 132, and 142 each have two feature maps/channels, then the compression logic 108 configures the compressed spatial map sets 120, 130, 140, and 150 to also have two feature maps/channels. Similarly, if the respective input image patches 112, 122, 132, and 142 each have three feature maps/channels, then the compression logic 108 configures the compressed spatial map sets 120, 130, 140, and 150 to also have three feature maps/channels. In the same vein, if the respective input image patches 112, 122, 132, and 142 each have four feature maps/channels, then the compression logic 108 configures the compressed spatial map sets 120, 130, 140, and 150 to also have four feature maps/channels.

[0167] Figure 2A shows that, during the second iteration of base calling, spatial maps 226 and corresponding compressed spatial maps 230 are generated only for the non-overlapping sequencing cycle 6 by processing input image data 222 through the spatial logic 104 and the compression logic 108. Accordingly, the input image patches 112, 122, 132, and 142 for the overlapping cycles 2, 3, 4, and 5 (highlighted with grey fill in the legend) are not reprocessed to avoid redundant convolutions.

[0168] Figure 2B shows that the compressed spatial map sets 120, 130, 140, and 150 generated during the first iteration of base calling are used in conjunction with the compressed spatial map set 230 generated during the second iteration of base calling to generate the base calls 292 for the center sequencing cycle 4. In Figure 2B, temporal map sets 174, 176, and 278 are generated by the first temporal convolution layer of the temporal network 160 in a manner

similar to the one discussed above with respect to Figure 1B. Temporal map set 282 is generated by the second and last temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Figure 2C shows that the output layer 190 processes the final temporal map set 282 generated during the second iteration of base calling and produces the base calls 292 for the center sequencing cycle 4.

[0169] The third window of sequencing cycles includes sequencing cycles 3, 4, 5, 6, and 7 and a third iteration of base calling produces base calls 392 for the center sequencing cycle 5. Accordingly, sequencing cycles 3, 4, 5, and 6 are the overlapping sequencing cycles between the second and third windows or between the second and third iterations of base calling.

[0170] The disclosed base calling systems and techniques store, in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM), the compressed spatial map sets 130, 140, and 150 generated during the first iteration of base calling for the respective sequencing cycles 3, 4, and 5 and the compressed spatial map set 230 generated during the second iteration of base calling for the sequencing cycle 6. During the third iteration of base calling, the disclosed base calling systems and techniques do not reprocess the respective input image patches 122, 132, 142, and 222 for the overlapping cycles 3, 4, 5, and 6 through the spatial network 104. Instead, during the third iteration of base calling, the disclosed base calling systems and techniques reuse the previously generated compressed spatial map sets 130, 140, 150, and 230 in lieu of the respective input image patches 122, 132, 142, and 222.

[0171] Figure 3A shows that, during the third iteration of base calling, spatial maps 326 and corresponding compressed spatial maps 330 are generated only for the non-overlapping sequencing cycle 7 by processing input image data 322 through the spatial logic 104 and the compression logic 108. Accordingly, the input image patches 122, 132, 142, and 222 for the overlapping cycles 3, 4, 5, and 6 (highlighted with grey fill in the legend) are not reprocessed to avoid redundant convolutions.

[0172] Figure 3B shows that the compressed spatial map sets 130, 140, 150, and 230 generated during the first and second iterations of base calling are used in conjunction with the compressed spatial map set 330 generated during the third iteration of base calling to generate the base calls 392 for the center sequencing cycle 5. In Figure 3B, temporal map sets 176, 278, and 378 are generated by the first temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Temporal map set 382 is generated by the second and last temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Figure 3C shows that the output layer 190 processes the final temporal map set 382 generated during the third iteration of base calling and produces the base calls 392 for the center sequencing cycle 5.

[0173] A compressed spatial map set once generated for a given sequencing cycle can be reused for base calling any subsequent sequencing cycle. Figure 4A shows a fourteenth iteration of base calling for base calling a center sequencing cycle 16. Figure 4B shows that compressed spatial maps previously generated for sequencing cycles 1 to 29 are used to generate a final temporal map set 482 for base calling the center sequencing cycle 16. Figure 4C shows that the output layer 190 processes the final temporal map set 482 generated during the fourteenth iteration of base calling and produces base calls 492 for the center sequencing cycle 16.

[0174] Figure 5A illustrates one implementation of filtering the compressed spatial map sets 110, 120, 130, 140, and 150 using filtering logic 502 to generate respective compressed, filtered spatial maps 510, 520, 530, 540, and 550 (depicting only reliable clusters) for the respective sequencing cycles 1, 2, 3, 4, and 5 during the first iteration of base calling. As discussed above, the unreliable clusters data 128 identifies those portions (*e.g.*, pixels) of the spatial maps and the compressed spatial maps that correspond to the unreliable clusters. Such pixels can be identified, for example, based on location coordinates of the unreliable clusters.

[0175] The filtering logic 502 uses the data 128 identifying the unreliable clusters to filter out (or discard or remove) those pixels from the compressed spatial map sets 110, 120, 130, 140, and 150 that correspond to (*i.e.*, depict intensity emissions of) the unreliable clusters. In some implementations, this results in 75% of pixels being discarded from the compressed spatial map sets, and thereby prevents many unproductive convolutions.

[0176] In Figure 5A, filtered temporal map sets 572, 574, and 576 (depicting only reliable clusters) are generated from the compressed, filtered spatial maps 510, 520, 530, 540, and 550 for base calling the center sequencing cycle 3. The filtered temporal map sets 572, 574, and 576 (depicting only reliable clusters) are generated by the first temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Filtered temporal map set 582 (depicting only reliable clusters) is generated by the second and last temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Figure 5B shows that the output layer 190 processes the final filtered temporal map set 582 generated during the first iteration of base calling and produces base calls 592 for the center sequencing cycle 3.

[0177] Figure 6A illustrates one implementation of filtering the compressed spatial map sets 120, 130, 140, 150, and 230 using the filtering logic 502 to generate respective compressed, filtered spatial maps 520, 530, 540, 550, and 650 (depicting only reliable clusters) for the respective sequencing cycles 2, 3, 4, 5, and 6 during the second iteration of base calling. The filtering logic 502 uses the data 128 identifying the unreliable clusters to filter out (or discard or

remove) those pixels from the compressed spatial map sets 120, 130, 140, 150, and 230 that correspond to (*i.e.*, depict intensity emissions of) the unreliable clusters.

[0178] In Figure 6A, filtered temporal map sets 574, 576, and 676 (depicting only reliable clusters) are generated from the compressed, filtered spatial maps 520, 530, 540, 550, and 650 for base calling the center sequencing cycle 4. The filtered temporal map sets 574, 576, and 676 (depicting only reliable clusters) are generated by the first temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Filtered temporal map set 682 (depicting only reliable clusters) is generated by the second and last temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Figure 6B shows that the output layer 190 processes the final filtered temporal map set 682 generated during the second iteration of base calling and produces base calls 692 for the center sequencing cycle 4.

[0179] Figure 7A illustrates one implementation of filtering the compressed spatial map sets 130, 140, 150, 230, and 330 using the filtering logic 502 to generate respective compressed, filtered spatial maps 530, 540, 550, 650, and 750 (depicting only reliable clusters) for the respective sequencing cycles 3, 4, 5, 6 and 7 during the third iteration of base calling. The filtering logic 502 uses the data 128 identifying the unreliable clusters to filter out (or discard or remove) those pixels from the compressed spatial map sets 130, 140, 150, 230, and 330 that correspond to (*i.e.*, depict intensity emissions of) the unreliable clusters.

[0180] In Figure 7A, filtered temporal map sets 576, 676, and 776 (depicting only reliable clusters) are generated from the compressed, filtered spatial maps 530, 540, 550, 650, 750 for base calling the center sequencing cycle 5. The filtered temporal map sets 576, 676, and 776 (depicting only reliable clusters) are generated by the first temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Filtered temporal map set 782 (depicting only reliable clusters) is generated by the second and last temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figure 1B. Figure 7B shows that the output layer 190 processes the final filtered temporal map set 782 generated during the third iteration of base calling and produces base calls 792 for the center sequencing cycle 5.

[0181] In other implementations, the compression logic 108 can configure the corresponding compressed temporal map sets to each have more than four feature maps.

[0182] The compression logic 108 discussed above with respect to the spatial feature maps applies equivalently to compression of temporal feature maps generated by the temporal logic 160. The reusing of once generated compressed spatial feature maps in subsequent sequencing

cycles also applies equivalently to reusing of once generated compressed temporal feature maps in subsequent sequencing cycles.

[0183] In some implementations, reusing the compressed temporal feature maps results in two orders of efficiency and compute savings over reusing the compressed spatial feature maps because the compressed temporal feature maps are generated from the compressed spatial feature maps at a later stage of the processing pipeline. Repurposing the intermediate results from a further processing engine (*i.e.*, the temporal network 160) increases the number of earlier processing steps that can be skipped. That is, reusing the compressed spatial feature maps eliminates redundant processing of the original image data through the spatial network 104 but can include redundantly processing of the compressed spatial feature maps through the temporal network 160. In contrast, reusing the compressed temporal feature maps eliminates both—the redundant processing of original image data through the spatial network 104 and the redundant processing of the compressed spatial feature maps through the temporal network 160.

[0184] Figure 8A shows one implementation of processing the sets of temporal feature maps 172, 174, and 176 generated during the first iteration of base calling through the compression logic 108 to generate respective sets of compressed temporal feature maps 802, 804, and 806. The compressed temporal feature map sets 802, 804, and 806 are generated by the compression logic 108 in a manner similar to the one discussed above with respect to Figures 1E and 1F. That is, for example, if the first temporal feature map set 172 has, for example, twenty-one feature maps (or channels or depth = 21), then the compression logic 108 can configure the corresponding compressed temporal feature map set 802 to have one, two, three, four, or feature maps. The sets of compressed temporal feature maps 802, 804, and 806 are processed by the second temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed temporal feature map set 814. Figure 8B shows that the output layer 190 processes the final compressed temporal feature map set 814 generated during the first iteration of base calling and produces base calls 892 for the center sequencing cycle 3.

[0185] Figure 9A shows one implementation of reusing, in the second base calling iteration, those compressed temporal maps that were generated in the first base calling iteration. That is, the first and second sets of compressed temporal maps 804 and 806 were generated in Figure 8A for the first base calling iteration and are now repurposed in the second base calling iteration shown in Figures 9A and 9B.

[0186] Note that the first and second sets of compressed temporal maps 804 and 806 were generated in Figure 8A from the first and second sets of temporal maps 172 and 174. Further note that the first and second sets of temporal maps 172 and 174 were generated in Figure 1B

from the compressed spatial maps 110, 120, 130, and 140, which in turn were generated in Figure 1A from the respective spatial maps 106, 116, 126, and 136, which in turn were generated in Figure 1A from the respective image patches 102, 112, 122, and 132.

[0187] Unlike Figures 1B, 2B, and 3B that redundantly generate overlapping temporal maps 172, 174, and 176, in Figure 9A, the overlapping temporal maps 174 and 176 (depicted in Figure 9A with dotted lines and ghost text) are not redundantly generated from Figure 8A (the first base calling iteration) to Figure 9A (the second base calling iteration). This occurs because the compression logic 108 is incorporated into the temporal network 160 to generate the first and second sets of compressed temporal maps 804 and 806 in the first base calling iteration, which replace the overlapping temporal maps 174 and 176 in the second base calling iteration. The compressed temporal maps can be stored in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM).

[0188] Figure 9A also shows processing the non-overlapping temporal maps 278 (*i.e.*, non-overlapping between the first and second base calling iterations) through the compression logic 108 to generate compressed temporal maps 906. The compressed temporal map sets 804, 806, and 906 are processed by the second temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed temporal feature map set 914. Figure 9B shows that the output layer 190 processes the final compressed temporal feature map set 914 generated during the second iteration of base calling and produces base calls 992 for the center sequencing cycle 4.

[0189] Unlike Figures 1B, 2B, and 3B that redundantly generate overlapping temporal maps 174, 176, and 278, in Figure 10A, the overlapping temporal maps 176 and 278 (depicted in Figure 10A with dotted lines and ghost text) are not redundantly generated from Figure 9A (the second base calling iteration) to Figure 10A (the third base calling iteration). This occurs because the compression logic 108 is incorporated into the temporal network 160 to generate the first and second sets of compressed temporal maps 806 and 906 in the first and second base calling iterations, which replace the overlapping temporal maps 176 and 278 in the third base calling iteration. The compressed temporal maps can be stored in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM).

[0190] Figure 10A also shows processing the non-overlapping temporal maps 378 (*i.e.*, non-overlapping between the second and third base calling iterations) through the compression logic 108 to generate compressed temporal maps 1006. The compressed temporal map sets 806, 906, and 1006 are processed by the second temporal convolution layer of the temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed temporal feature map set 1014. Figure 10B shows that the output layer 190

processes the final compressed temporal feature map set 1014 generated during the third iteration of base calling and produces base calls 1092 for the center sequencing cycle 5.

[0191] Figure 11A shows one implementation of processing the sets of filtered temporal feature maps 572, 574, and 576 generated during the first iteration of base calling through the compression logic 108 to generate respective sets of compressed, filtered temporal feature maps 1102, 1104, and 1306. The compressed, filtered temporal feature map sets 1102, 1104, and 1106 (depicting only reliable clusters) are generated by the compression logic 108 in a manner similar to the one discussed above with respect to Figures 1E and 1F. That is, for example, if the first filtered temporal feature map set 572 has, for example, twenty-one feature maps (or channels or depth = 21), then the compression logic 108 can configure the corresponding compressed, filtered temporal feature map set 1102 to have one, two, three, or four feature maps. The sets of compressed, filtered temporal feature maps 1102, 1104, and 1106 are processed by the second filtered temporal convolution layer of the filtered temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed, filtered temporal feature map set 1114. Figure 8B shows that the output layer 190 processes the final compressed, filtered temporal feature map set 1114 generated during the first iteration of base calling and produces base calls 1192 for the center sequencing cycle 3.

[0192] In other implementations, the compression logic 108 can configure the corresponding compressed feature map sets to each have more than four feature maps.

[0193] Figure 12A shows one implementation of reusing, in the second base calling iteration, those compressed, filtered temporal maps that were generated in the first base calling iteration. That is, the first and second sets of compressed, filtered temporal maps 1104 and 1106 were generated in Figure 11A for the first base calling iteration and are now repurposed in the second base calling iteration shown in Figures 12A and 2B.

[0194] Note that the first and second sets of compressed, filtered temporal maps 1104 and 1106 were generated in Figure 11A from the first and second sets of filtered temporal maps 572 and 574. Further note that the first and second sets of filtered temporal maps 572 and 574 were generated in Figure 5A from the compressed, filtered spatial maps 510, 520, 530, and 540, which in turn were generated in Figure 5A from the respective compressed spatial maps 110, 120, 130, and 140, which in turn were generated in Figure 1A from the respective spatial maps 106, 116, 126, and 136, which in turn were generated in Figure 1A from the respective image patches 102, 112, 122, and 132.

[0195] Unlike Figures 5A, 6A, and 7A that redundantly generate overlapping filtered temporal maps 572, 574, and 576, in Figure12A, the overlapping filtered temporal maps 574 and 576 (depicted in Figure 12A with dotted lines and ghost text) are not redundantly generated from

Figure 11A (the first base calling iteration) to Figure 12A (the second base calling iteration). This occurs because the compression logic 108 is incorporated into the filtered temporal network 160 to generate the first and second sets of compressed, filtered temporal maps 1104 and 1106 in the first base calling iteration, which replace the overlapping filtered temporal maps 574 and 576 in the second base calling iteration. The compressed, filtered temporal maps can be stored in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM).

[0196] Figure 12A also shows processing the non-overlapping filtered temporal maps 676 (*i.e.*, non-overlapping between the first and second base calling iterations) through the compression logic 108 to generate compressed, filtered temporal maps 1206 (depicting only reliable clusters). The compressed, filtered temporal map sets 1104, 1106, and 1206 are processed by the second filtered temporal convolution layer of the filtered temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed, filtered temporal feature map set 1214. Figure 12B shows that the output layer 190 processes the final compressed, filtered temporal feature map set 1214 generated during the second iteration of base calling and produces base calls 1292 for the center sequencing cycle 4.

[0197] Unlike Figures 5A, 6A, and 7A that redundantly generate overlapping filtered temporal maps 574, 576, and 676, in Figure 13A, the overlapping filtered temporal maps 576 and 676 (depicted in Figure 13A with dotted lines and ghost text) are not redundantly generated from Figure 12A (the second base calling iteration) to Figure 13A (the third base calling iteration). This occurs because the compression logic 108 is incorporated into the filtered temporal network 160 to generate the first and second sets of compressed, filtered temporal maps 1106 and 1206 in the first and second base calling iterations, which replace the overlapping filtered temporal maps 576 and 676 in the third base calling iteration. The compressed, filtered temporal maps can be stored in memory (*e.g.*, on-chip DRM, on-chip SRAM or BRAM, off-chip DRAM).

[0198] Figure 13A also shows processing the non-overlapping filtered temporal maps 776 (*i.e.*, non-overlapping between the second and third base calling iterations) through the compression logic 108 to generate compressed, filtered temporal maps 1306 (depicting only reliable clusters). The compressed, filtered temporal map sets 1106, 1206, and 1306 are processed by the second filtered temporal convolution layer of the filtered temporal network 160 in a manner similar to the one discussed above with respect to Figures 1B to generate a final compressed, filtered temporal feature map set 1314. Figure 13B shows that the output layer 190 processes the final compressed, filtered temporal feature map set 1314 generated during the third iteration of base calling and produces base calls 1392 for the center sequencing cycle 5.

[0199] Figure 14 illustrates a first example architecture of the neural network-based base caller 100. In the illustrated implementation, the neural network-based base caller 100 comprises

the spatial network 104, the compression network 108, and the temporal network 160. The spatial network 104 comprises seven spatial convolution layers. The compression network 108 comprises a compression layer. The temporal network 160 comprises two temporal convolution layers.

[0200] Each of the seven spatial convolution layers can have a same number of convolution filters or can have a different number of convolution filters. The first spatial convolution layer can have S1 number of filters, where S1 can be, for example, 7, 14, 21, 64, 128, or 254. The second spatial convolution layer can have S2 number of filters, where S2 can be, for example, 7, 14, 21, 64, 128, or 254. The third spatial convolution layer can have S3 number of filters, where S3 can be, for example, 7, 14, 21, 64, 128, or 254. The fourth spatial convolution layer can have S4 number of filters, where S4 can be, for example, 7, 14, 21, 64, 128, or 254. The fifth spatial convolution layer can have S5 number of filters, where S5 can be, for example, 7, 14, 21, 64, 128, or 254. The sixth spatial convolution layer can have S6 number of filters, where S6 can be, for example, 7, 14, 21, 64, 128, or 254. The seventh spatial convolution layer can have S7 number of filters, where S7 can be, for example, 7, 14, 21, 64, 128, or 254.

[0201] The compression layer can have C1 number of filters, where C1 can be, for example, 1, 2, 3, 4, or more.

[0202] Each of the two temporal convolution layers can have a same number of convolution filters or can have a different number of convolution filters. The first temporal convolution layer can have T1 number of filters, where T1 can be, for example, 7, 14, 21, 64, 128, or 254. The second temporal convolution layer can have T2 number of filters, where T2 can be, for example, 7, 14, 21, 64, 128, or 254. Figure 14 also shows feature maps 1412 generated by each of the layers of the neural network-based base caller 100.

[0203] Figure 15 illustrates a second example architecture of the neural network-based base caller 100. Figure 15 shows the filtering logic 502 as part of the neural network-based base caller 100. In other implementations, the filtering logic 502 is not part of the neural network-based base caller 100. The compressed feature maps C1 have a spatial dimensionality of P1 x P2. The filtering logic 502 filters out those pixels in the compressed feature maps C1 that correspond to unreliable clusters, and generates compressed, filtered feature maps F1 with the spatial dimensionality of P3 x P4. The compressed, filtered feature maps F1 depict only reliable clusters. In one implementation, the filtering logic 502 discards 75% of the pixels in the compressed feature maps C1 and therefore P3 is 25% of P1 and P4 is 25% of P2. Figure 15 also shows feature maps 1512 generated by each of the layers of the neural network-based base caller 100.

[0204] Figure 16 illustrates a third example architecture of the neural network-based base caller 100. Figure 16 shows that the compression network 108 is used to compress outputs of the spatial network 104 as well as the temporal network 160. Figure 16 also shows feature maps 1612 generated by each of the layers of the neural network-based base caller 100.

[0205] Figure 17 illustrates a fourth example architecture of the neural network-based base caller 100. Figure 17 shows that the filtering logic 502 is applied to the compressed outputs of the spatial network 104 to generate compressed and filtered temporal outputs from the temporal network 160. Figure 17 also shows feature maps 1712 generated by each of the layers of the neural network-based base caller 100.

[0206] Figure 18 shows one implementation of a filter configuration logic 1804 that configures a count (or numerosity) of convolution filters in the compression layer 108 in dependence upon a number of channels in the input data. This allows the compressed feature maps to be lossless representatives of the input data. In some implementations, the input data can be overwritten in memory with the corresponding compressed representation for reuse in subsequent sequencing cycles.

[0207] In one implementation, for input data that contains only one channel 1812 in each per-cycle input (*e.g.*, only one image channel), the filter configuration logic 1804 configures the compression layer 108 with only one convolution filter 1816 that generates only one compressed feature map 1818 per-sequencing cycle. In another implementation, for input data that contains two channels 1822 in each per-cycle input (*e.g.*, two image channels like blue and green image channels in the sequencing images corresponding to blue and green lasers), the filter configuration logic 1804 configures the compression layer 108 with two convolution filters 1826 that generate two compressed feature maps 1828 per-sequencing cycle. In yet another implementation, for input data that contains three channels 1832 in each per-cycle input (*e.g.*, three image channels), the filter configuration logic 1804 configures the compression layer 108 with three convolution filters 1836 that generate three compressed feature maps 1838 per-sequencing cycle. In yet further implementation, for input data that contains four channels 1842 in each per-cycle input (*e.g.*, four image channels like A, C, T, and G channels in the sequencing images corresponding to the nucleotides A, C, T, and G), the filter configuration logic 1804 configures the compression layer 108 with four convolution filters 1846 that generate four compressed feature maps 1848 per-sequencing cycle. In other implementations, the compression logic 108 can configure the corresponding compressed feature map sets to each have more than four feature maps, and therefore select more than four filters for the compression layer 108.

[0208] 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."

[0209] The sequencing system 1900A can operate to obtain any information or data that relates to at least one 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.

[0210] 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.

[0211] 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.

[0212] 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.

[0213] 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.

[0214] 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 also 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.

[0215] 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 array (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 at least one of obtain and 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.

[0216] 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.

[0217] 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, 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 cloud).

[0218] 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.

[0219] 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.

[0220] 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.

[0221] 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 removably engaged with a bioassay system via a cartridge or other biosensor.

[0222] 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 of 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 of 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.

[0223] The system receptacle or interface 1910 is configured to engage the biosensor 1912 in at least one of a mechanical, electrical, and 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 removably coupled to the system receptacle 1910 in a mechanical manner, in an electrical manner, and also in a fluidic manner.

[0224] 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.

[0225] 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.

[0226] 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.

[0227] 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.

[0228] The plurality of modules 1926-1948 include system modules 1928-1932, 1926 that communicate with the sub-systems 1908, 1914, 1904, and 1916, respectively. 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.

[0229] 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.

[0230] The plurality of modules 1926-1948 also includes an analysis module 1944 (also called signal processing module or signal processor) that receives and analyzes the signal data (*e.g.*, image data) from the biosensor 1912. 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.

[0231] 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 and process the image data for each of the plurality of sequencing cycles through the neural network-based base caller 100 and produce a base call for at least some of the analytes at each of the plurality of sequencing cycle. 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).

[0232] 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.

[0233] 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, 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 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 7,0197,026; WO 91/066719; WO 07/123744; US 7,329,492; US 7,211,414; US 7,3119,019; US 7,4019,2191, and US 20019/01470190192, each of which is incorporated herein by reference.

[0234] 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.

[0235] 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 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.

[0236] 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 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.

[0237] 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.

[0238] 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, 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. 2007/01667019 A1, US Patent Application Publication No. 2006/01196*3901 A1, US Patent No. 7,0197,026, US Patent Application Publication No. 2006/0240439 A1, US Patent Application Publication No. 2006/021914714709 A1, PCT Publication No. WO 019/0619514, US Patent Application Publication No. 20019/014700900 A1, PCT Publication No. WO 06/019B199 and PCT Publication No. WO 07/014702191, each of which is incorporated herein by reference in its entirety. Exemplary reagents for reversible terminator-based SBS are described in US 7,1941,444; US 7,0197,026; US 7,414,14716; US 7,427,673; US 7,1966,1937; US 7,1992,4319 and WO 07/14193193619, each of which is incorporated herein by reference in its entirety.

[0239] In some implementations, the amplification and SBS modules may operate in a single assay protocol where, for example, template nucleic acid is amplified and subsequently sequenced within the same cartridge.

[0240] 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.

[0241] 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 to the row-wise and/or column-wise location of the sensors on an array of sensors.

[0242] Figure 19C is a simplified block diagram of a system for 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 100) 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 in during a base call operation.

[0243] The system in this example includes the CPU 1952, which executes a runtime program/logic 1980 to coordinate the base call operations, 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 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 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.

[0244] 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 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 of working data used in the base call operation. The bus 1993 can also be implemented using a high throughput technology, such as bus technology compatible with the PCIe standards.

[0245] 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. 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.

[0246] Implementations described herein implement the neural network-based base caller 100 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.

[0247] Alternatives for the configurable processor 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.

[0248] 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.

[0249] 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 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.

[0250] Also, the configurable processor 1946 is configured with data flow logic 1997 to execute the neural network-based base caller 100. The logic 1997 comprises multi-cycle execution clusters (*e.g.*, 1979) which, in this example, includes 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.

[0251] 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 execution of a run of the neural network-based base caller 100 to the available execution clusters, readiness to provide trained parameters for the neural network-based base caller 100, readiness to provide output patches of base call classification data, and other control data used for execution of the neural network-based base caller 100.

[0252] The configurable processor 1946 is configured to execute runs of the neural network-based base caller 100 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 100 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 100 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 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 time sequence. Examples are described herein in which the number N is an integer equal to or greater than five.

[0253] 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 100, 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.

[0254] 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 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.

[0255] During execution of the neural network-based base caller 100 as described below, tile data can also include data produced during execution of the neural network-based base caller 100, referred to as intermediate data, which can be reused rather than recomputed during a run of the neural network-based base caller 100. For example, during execution of the neural network-based base caller 100, 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.

[0256] As illustrated, a system is described for analysis of base call sensor output, comprising 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 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.

[0257] 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.

[0258] 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 are provided on lines 2000 to 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, and can 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 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 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.

[0259] 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.

[0260] 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.*, memory 1448A) via 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 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 line 2083 to the allocated cluster 2085. The wrapper provides trained parameters, such as weights and biases on line 2081 to the cluster 2085 retrieved from the on-board DRAM 2099. The wrapper provides configuration and control data on 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 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.

[0261] 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.

[0262] 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 cycle number. A base calling operation can comprise, for a representative example, on the order of hundreds of sensing cycles. 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.

[0263] 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 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 other type of metadata about the sensor data and the tiles.

[0264] 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.

[0265] 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 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 memory management logic in the wrapper 2090. The runtime program can control the sequencing operations to complete analysis of all the arrays of tile data for all the cycles in the run in a continuous flow to provide real time analysis.

[0266] Figure 21 illustrates another implementation of the disclosed data flow logic making compressed spatial maps, generated during the first base calling iteration, available during the second base calling iteration from off-chip memory 2116 (*e.g.*, off-chip DRAM, host RAM, host high bandwidth memory (HBM)).

[0267] In one implementation, a host memory (*e.g.*, memory 1948B) attached to a host processor (*e.g.*, CPU 1952) is configured to receive a progression of sequencing images 2102 as a sequencing run progresses. A configurable processor (*e.g.*, configurable processor 1946) has an array of processing units. Processing units in the array of processing units are configured to execute the neural network-based base caller 100 to produce base call predictions. The data flow logic 1997 has access to the host memory, the host processor, and the configurable processor. For the first base calling iteration, the data flow logic 1997 loads sequencing images for

sequencing cycles in the first window of sequencing cycles (*e.g.*, the sequencing cycles 1 to 5 in Figure 1A) on the configurable processor from the host memory.

[0268] The runtime logic 1980 is configured to cause the processing units of the configurable processor to execute the spatial network 104 of the neural network-based base caller 100 on the sequencing images 2102 on a cycle-by-cycle basis and generate spatial feature map sets 2106 for each of the sequencing cycles in the first window of sequencing cycles. In one implementation, the runtime logic 1980 executes, in parallel, multiple processing clusters of the neural network-based base caller 100 on patches 2104 tiled from the sequencing images 2102. The multiple processing clusters apply the spatial network 104 on the patches 2104 on a patch-by-patch basis 2105.

[0269] The runtime logic 1980 is configured to cause the processing units of the configurable processor to execute the compression network 108 of the neural network-based base caller 100 on the spatial feature map sets 2106 on the cycle-by-cycle basis and generate compressed spatial feature map sets 2107, and process the compressed spatial feature maps sets 2107 through the temporal network 160 and the output network 190 to produce base call predications 2111 for one or more sequencing cycles in the first window of sequencing cycles. The temporal network 160 generates the temporal feature maps 2108. The output network 190 generates base call classification scores 2110 (*e.g.*, unnormalized base-wise scores). In one implementation, the compressed spatial feature map sets 2107 are stored on the off-chip memory 2116.

[0270] In one implementation, the data flow logic 1997 is configured to move the compressed spatial feature map sets 2107 to the host memory 2116 and overwrite corresponding ones of the sequencing images 2102 with the compressed spatial feature map sets 2107. In other implementations, corresponding ones of the patches 2104 are replaced by the compressed spatial feature map sets 2107.

[0271] For the second base calling iteration and for the second window of sequencing cycles (*e.g.*, the sequencing cycles 2 to 6 in Figure 2A) that shares one or more overlapping sequencing cycles (*e.g.*, the sequencing cycles 2-5) with the first window of sequencing cycle, and has at least one non-overlapping sequencing cycle (*e.g.*, the sequencing cycle 6), the data flow logic 1997 is configured to load, on the configurable processor from the host memory, compressed spatial feature map sets 2126 for the overlapping sequencing cycles, and sequencing images 2122 (or patches 2124) for the non-overlapping sequencing cycle.

[0272] The runtime logic 1980 is configured to cause the processing units of the configurable processor to execute the spatial network104 on the sequencing images 2122 for the non-overlapping sequencing cycle and generate a spatial feature map set 2126 for the non-

overlapping sequencing cycle. In one implementation, the multiple processing clusters apply the spatial network 104 on the patches 2124 on a patch-by-patch basis 2125.

[0273] The runtime logic 1980 is configured to cause the processing units of the configurable processor to execute the compression network 108 on the spatial feature map set 2126 and generate a compressed spatial feature map set 2127 for the non-overlapping sequencing cycle, and process the compressed spatial feature maps sets 2126 for the overlapping sequencing cycles and the compressed spatial feature map set 2127 for the non-overlapping sequencing cycle through the temporal network 160 and the output network 190 to produce base call predications 2131 for one or more sequencing cycles in the second window of sequencing cycles. The temporal network 160 generates the temporal feature maps 2128. The output network 190 generates base call classification scores 2129 (*e.g.*, unnormalized base-wise scores). In one implementation, the compressed spatial feature map set 2127 is stored on the off-chip memory 2116.

[0274] Figure 22 illustrates one implementation of a disclosed data flow logic making compressed spatial maps, generated during the first base calling iteration, available during the second base calling iteration from on-chip memory 2216 (*e.g.*, processor memory like on-chip DRAM, on-chip SRAM, on-chip BRAM, DRAM attached to the processor via an interconnect). In Figure 22, the compressed spatial feature maps sets 2107 and the compressed spatial feature map set 2127 are stored on the on-chip memory 2216. Also in Figure 22, the data flow logic 1997 is configured to load, on the configurable processor from the on-chip memory 2216, the compressed spatial feature map sets 2126 for the overlapping sequencing cycles.

Split Architecture

[0275] Figure 23 illustrates one implementation of a so-called split architecture of the neural network-based base caller 100. As discussed above, the spatial convolution network 104 is configured to process a window of per-cycle sequencing image sets for a series of sequencing cycles (cycles N+2, N+1, N, N-1, and N-2) of a sequencing run on a cycle-by-cycle basis by separately convolving respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective sequences 2301, 2302, 2303, 2304, and 2405 of spatial convolution layers to generate respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles. For example, each of the five sequences 2301, 2302, 2303, 2304, and 2405 of spatial convolution layers has seven spatial convolution layers (*i.e.*, layers L1 to L7 in Figure 23).

[0276] The respective sequences 2301, 2302, 2303, 2304, and 2405 of spatial convolution layers have respective sequences of spatial convolution filter banks (*e.g.*, the sequence

comprising spatial convolution filter banks 2310, 2311, 2312, 2313, 2314, 2315, and 2316 for the sequence 2301 of spatial convolution layers). In one implementation, trained coefficients (or weights) of spatial convolution filters in spatial convolution filter banks of the respective sequences of spatial convolution filter banks vary between sequences of spatial convolution layers in the respective sequences of spatial convolution layers.

[0277] For example, the spatial convolution layer sequences 2301, 2302, 2303, 2304, and 2405 are configured with convolution filters with different trained coefficients. In another example, the convolution filters in corresponding level spatial convolution layers have different trained coefficients (*e.g.*, convolution filter banks 2382, 2383, 2384, 2385, and 2312 in the respective third spatial convolution layers of the five sequences 2301, 2302, 2303, 2304, and 2405 of spatial convolution layers).

[0278] The temporal convolution network 160 is configured to process the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups (*e.g.*, groups 2360, 2361, and 2362) of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks 2321, 2322, and 2323 of a first temporal convolution layer 2320 to generate respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets. In one implementation, trained coefficients (or weights) of temporal convolution filters in the respective temporal convolution filter banks vary between temporal convolution filter banks 2321, 2322, and 2323 in the respective temporal convolution filter banks.

Skip Architecture

[0279] Figure 24A depicts a residual (or skip) connection that reinjects prior information downstream via feature-map addition. A residual connection comprises reinjecting previous representations into the downstream flow of data by adding a past output tensor to a later output tensor, which helps prevent information loss along the data processing flow. Residual connections tackle two common problems that plague any large-scale deep learning model: vanishing gradients and representational bottlenecks.

[0280] A residual connection comprises making the output of an earlier layer available as input to a later layer, effectively creating a shortcut in a sequential network. Rather than being concatenated to the later activation, the earlier output is summed with the later activation, which assumes that both activations are the same size. If they are of different sizes, a linear transformation to reshape the earlier activation into the target shape can be used.

[0281] Figure 24B depicts one implementation of residual blocks and skip connections. A residual network stacks a number of residual units to alleviate the degradation of training

accuracy. Residual blocks make use of special additive skip connections to combat vanishing gradients in deep neural networks. At the beginning of a residual block, the data flow is separated into two streams: the first carries the unchanged input of the block, while the second applies weights and non-linearities. At the end of the block, the two streams are merged using an element-wise sum. The main advantage of such constructs is to allow the gradient to flow through the network more easily.

[0282] Configured with a residual network, in some implementations, the neural network-based base caller 100 is easily trained and improved accuracy can be achieved for image classification and object detection. The neural network-based base caller 100 connects the output of the l^{th} layer as input to the $(l+1)^{th}$ layer, which gives rise to the following layer transition: $x_l = H_l(x_{l-1})$. Residual blocks add a skip connection that bypasses the non-linear transformations with an identify function: $x_l = H_l(x_{l-1}) + x_{l-1}$. An advantage of residual blocks is that the gradient can flow directly through the identity function from later layers to the earlier layers (*e.g.*, spatial, and temporal convolution layers). The identity function and the output of H_l are combined by summation (addition).

[0283] Figure 24C shows a residual architecture of the neural network-based base caller 100 in which the spatial convolution layers are grouped into residual blocks with skip connections. In other implementations, the temporal convolution layers of the neural network-based base caller 100 are grouped into residual blocks with skip connections.

[0284] In the implementation illustrated in Figure 24C, the second and third spatial convolution layers are grouped into a first residual block 2412; the fourth and fifth spatial convolution layers are grouped into a second residual block 2422; and the sixth and seventh spatial convolution layers are grouped into a third residual block 2432.

[0285] Figure 25A shows details of a bus network of the neural network-based base caller 100. In one implementation, a given residual block 2585 of the bus network comprises a set of spatial convolution layers 2590 and 2592. A first spatial convolution layer 2590 in the set of spatial convolution layers receives as input a preceding output 2586 generated by a preceding spatial convolution layer not part of the given residual block 2585 (*e.g.*, a zero spatial convolution layer preceding the first spatial convolution layer 2590 in the spatial network 104). The first spatial convolution layer 2590 processes the preceding output 2586 and generates a first output 2591. A second spatial convolution layer 2592 in the set of spatial convolution layers, which succeeds the first spatial convolution layer 2590, receives the first output 2591, processes the first output 2591 and generates a second output 2593. In one implementation, the first spatial convolution layer 2590 has a non-linear activation function like ReLU that generates the first

output 2591. In another implementation, the second spatial convolution layer 2592 lacks the non-linear activation function.

[0286] A skip connection 2589 provides the preceding output 2586 to a summer 2594. The summer 2594 also receives the second output 2593 from the second spatial convolution layer 2592. The summer 2594 combines the preceding output 2586 and the second output 2593 and generates a summed output 2595. The summed output 2595 is further processed through a non-linear activation like ReLU to generate a final summed output 2587. The final summed output 2587 is then fed as input to a succeeding residual block, in some implementations. In some implementations, the preceding output 2586 is modified to be dimensionality-compatible with the second output 2593. For example, edges of feature maps in the preceding output 2586 are trimmed to produce feature maps that have the same spatial dimensionality as the feature maps in the second output 2593.

[0287] Figure 25B shows an example operation of the disclosed bus network. In one implementation, the bus network is configured to form buses (*e.g.*, 2516, 2526, 2536, 2602, 2604, 2702, and 2712) between spatial convolution layers within the respective sequences of spatial convolution layers. The buses are configured to cause respective per-cycle spatial feature map sets generated by two or more spatial convolution layers in a particular sequence of spatial convolution layer for a particular sequencing cycle to combine into a combined per-cycle spatial feature map set, and provide the combined per-cycle spatial feature map set as input to another spatial convolution layer in the particular sequence of spatial convolution layer.

[0288] For example, consider the first residual block 2412. Here, the two or more spatial convolution layers include a first spatial convolution layer and a third spatial convolution layer. The first spatial convolution layer generates a first per-cycle spatial feature map set 2520. The first spatial convolution layer provides the first per-cycle spatial feature map set 2520 as input to a second spatial convolution layer. The second spatial convolution layer processes the first per-cycle spatial feature map set 2520 and generates a second per-cycle spatial feature map set 2522. The second spatial convolution layer provides the second per-cycle spatial feature map set 2522 as input to the third spatial convolution layer. The third spatial convolution layer processes the second per-cycle spatial feature map set 2522 and generates a third per-cycle spatial feature map set 2524. The buses (*e.g.*, the skip bus 2519) are further configured to cause the first spatial feature map set 2520 and the third per-cycle spatial feature map set 2524 to combine (*e.g.*, summed or concatenated by a combiner 2502) into the combined per-cycle spatial feature map set 2518. Then, the another spatial convolution layer is a fourth spatial convolution layer that immediately succeeds the third spatial convolution layer in the particular sequence of spatial convolution layer. The fourth spatial convolution layer processes the combined per-cycle spatial

feature map set 2518 as input. The same notion analogously applies to the second and third residual blocks 2422 and 2432, where 2526 and 2536 are the skip buses like the skip buses 2516, and cause the respective combiners 2512 and 2532 to generate the respective combined per-cycle spatial feature map sets 2528 and 2538.

[0289] Figure 25C shows one implementation of a dimension compatibility logic 2532 that ensures that, prior to the combination, the incoming feature maps supplied by the skip buses are modified (*e.g.*, trimmed) to have the same spatial dimensionality as the receiving feature maps with which the incoming feature maps are combined by the combiners of the bus network.

[0290] Figure 26 shows another example of the disclosed bus network in which a skip bus 2602 combines the first spatial convolution layer's output with the output of the first residual block. Figure 26 also shows that a skip bus 2604 can also causes feature maps to be combined across residual blocks and across non-successive layers (*e.g.*, from layers 1 and 5) to be generate a combined representation that can be processed by another layer (*e.g.*, to layer 6).

[0291] Figure 27 shows yet another example of the disclosed bus network in which inputs and combined representations from multiple successive and/or non-successive layers (*e.g.*, from layer 1, combiner 2502, and combiner 2512) can be combined by example skip buses 2702, 2604, and 2712 to generate a combined representation that can be processed by another layer (*e.g.*, to layer 6).

[0292] Figure 28 shows one implementation of a scaling logic 2832 that scales the incoming feature maps supplied by the skip buses before combining them with the receiving feature maps with which they are combined by the combiners of the bus network. The values used by the scaling logic 2832 can range, for example, anywhere between 0 and 1, including 0 and 1. The scaling logic can be used, for example, to attenuate or amplify the strength/magnitude/value (*e.g.*, feature values (*e.g.*, floating point values) of the incoming feature maps.)

[0293] Figure 29 shows one implementation of skip connections between temporal convolution layers 2902, 2912, 2922, 2932, 2942, 2952, 2962, and 2972 of the temporal network 160. For example, a skip connection 2922 supplies temporal feature maps from the first temporal convolution layer 2902 to the third temporal convolution layer 2932.

[0294] Figure 30 compares base calling performance by the network network-based base caller 100 configured with the compression logic 108 (sqz2 base caller) against the network network-based base caller 100 without the compression logic 108 (used as baseline neural network models) and against Illumina's non-neural network-based base caller Real-Time Analysis (RTA) software) (use as a baseline traditional image processing model). As seen in the chart in Figure 30, the sqz2 base caller (purple fitted line) has a lower base calling error percentage ("Error %" on Y-axis) than the RTA base caller (black fitted line) and the two

instances of the network network-based base caller 100 without the compression logic 108 (red and cyan fitted lines).

[0295] Figure 31 shows savings in RAM and DRAM usage brought about by the use of the disclosed compression logic 108.

[0296] Figure 32 compares base calling performance by the network network-based base caller 100 configured with the split and skip architectures (split_res) against the RTA base caller and another version of the network network-based base caller 100 without the split and skip architectures (distilled). As seen in the chart in Figure 32, the split_res base caller (orange fitted line) has a lower base calling error percentage ("Error Count" on Y-axis) than the RTA base caller (blue fitted line).

[0297] "Logic" (*e.g.*, data flow logic), as used herein, can be implemented in the form of a computer product including a non-transitory computer readable storage medium with computer usable program code for performing the method steps described herein. The "logic" can be implemented in the form of an apparatus including a memory and at least one processor that is coupled to the memory and operative to perform exemplary method steps. The "logic" can be implemented in the form of means for carrying out one or more of the method steps described herein; the means can include (i) hardware module(s), (ii) software module(s) executing on one or more hardware processors, or (iii) a combination of hardware and software modules; any of (i)-(iii) implement the specific techniques set forth herein, and the software modules are stored in a computer readable storage medium (or multiple such media). In one implementation, the logic implements a data processing function. The logic can be a general purpose, single core or multicore, processor with a computer program specifying the function, a digital signal processor with a computer program, configurable logic such as an FPGA with a configuration file, a special purpose circuit such as a state machine, or any combination of these. Also, a computer program product can embody the computer program and configuration file portions of the logic.

[0298] Figure 33 is a computer system 3300 that can be used by the sequencing system 1900A to implement the base calling techniques disclosed herein. Computer system 3300 includes at least one central processing unit (CPU) 3372 that communicates with a number of peripheral devices via bus subsystem 3355. These peripheral devices can include a storage subsystem 3358 including, for example, memory devices and a file storage subsystem 3336, user interface input devices 3338, user interface output devices 3376, and a network interface subsystem 3374. The input and output devices allow user interaction with computer system 3300. Network interface subsystem 3374 provides an interface to outside networks, including an interface to corresponding interface devices in other computer systems.

[0299] In one implementation, the system controller 1906 is communicably linked to the storage subsystem 3310 and the user interface input devices 3338.

[0300] User interface input devices 3338 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 3300.

[0301] User interface output devices 3376 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 3300 to the user or to another machine or computer system.

[0302] Storage subsystem 3358 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 3378.

[0303] Deep learning processors 3378 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 3378 can be hosted by a deep learning cloud platform such as Google Cloud Platform™, Xilinx™, and Cirrascale™. Examples of deep learning processors 3378 include Google's Tensor Processing Unit (TPU)™, rackmount solutions like GX4 Rackmount Series™, GX33 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™, SambaNova's Reconfigurable Dataflow Unit (RDU)™, and others.

[0304] Memory subsystem 3322 used in the storage subsystem 3358 can include a number of memories including a main random access memory (RAM) 3332 for storage of instructions and data during program execution and a read only memory (ROM) 3334 in which fixed instructions are stored. A file storage subsystem 3336 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 3336 in the storage subsystem 3358, or in other machines accessible by the processor.

[0305] Bus subsystem 3355 provides a mechanism for letting the various components and subsystems of computer system 3300 communicate with each other as intended. Although bus subsystem 3355 is shown schematically as a single bus, alternative implementations of the bus subsystem can use multiple buses.

[0306] Computer system 3300 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 3300 depicted in Figure 33 is intended only as a specific example for purposes of illustrating the preferred implementations of the present invention. Many other configurations of computer system 3300 are possible having more or less components than the computer system depicted in Figure 33.

Clauses

[0307] We disclose the following clauses:

Compression (Squeeze)

1. An artificial intelligence-based method of base calling, the method including:

accessing a series of per-cycle analyte channel sets generated for sequencing cycles of a sequencing run;

processing, through a spatial network of a neural network-based base caller, a first window of per-cycle analyte channel sets in the series for a first window of sequencing cycles of the sequencing run, and generating respective sequences of spatial output sets for respective sequencing cycles in the first window of sequencing cycles;

processing, through a compression network of the neural network-based base caller, respective final spatial output sets in the respective sequences of spatial output sets, and generating respective compressed spatial output sets for the respective sequencing cycles in the first window of sequencing cycles; and

generating, based on the respective compressed spatial output sets, base call predictions for one or more sequencing cycles in the first window of sequencing cycles.

2. The artificial intelligence-based method of clause 1, wherein the respective final spatial output sets have M channels (feature maps), wherein the respective compressed spatial output sets have N channels (feature maps), and wherein $M > N$.

3. The artificial intelligence-based method of clause 1, further including:

for a second window of sequencing cycles of the sequencing run that shares, with the first window of sequencing cycles, one or more overlapping sequencing cycles for which the spatial network previously generated spatial output sets, and at least one non-overlapping sequencing cycle for which the spatial network is yet to generate a spatial output set,

processing, through the spatial network, a per-cycle analyte channel set only for the non-overlapping sequencing cycle, and generating a sequence of spatial output sets for the non-overlapping sequencing cycle, thereby bypassing reprocessing, through the spatial network, respective per-cycle analyte channel sets for the overlapping sequencing cycles;

processing, through the compression network, a final spatial output set in the sequence of spatial output sets, and generating a compressed spatial output set for the non-overlapping sequencing cycle, wherein the final spatial output has M channels (feature maps), wherein the compressed spatial output has N channels (feature maps), and wherein M > N; and

generating, based on respective compressed spatial output sets for the overlapping sequencing cycles previously generated for the first window of sequencing cycles and on the compressed spatial output set, base call predictions for one or more sequencing cycles in the second window of sequencing cycles, thereby substituting the respective compressed spatial output sets for the overlapping sequencing cycles for the respective per-cycle analyte channel sets for the overlapping sequencing cycles.

4. The artificial intelligence-based method of clause 3, further including:

for a third window of sequencing cycles of the sequencing run that shares, with the first and second windows of sequencing cycles, one or more overlapping sequencing cycles for which the spatial network previously generated spatial output sets, and at least one non-overlapping sequencing cycle for which the spatial network is yet to generate a spatial output set,

processing, through the spatial network, a per-cycle analyte channel set only for the non-overlapping sequencing cycle, and generating a sequence of spatial output sets for the non-overlapping sequencing cycle, thereby bypassing reprocessing, through the spatial network, respective per-cycle analyte channel sets for the overlapping sequencing cycles;

processing, through the compression network, a final spatial output set in the sequence of spatial output sets, and generating a compressed spatial output set for the non-overlapping sequencing cycle, wherein the final spatial output has M channels (feature maps), wherein the compressed spatial output has N channels (feature maps), and wherein M > N; and

generating, based on respective compressed spatial output sets for the overlapping sequencing cycles previously generated for the first and second windows of sequencing cycles and on the compressed spatial output set, base call predictions for one or more sequencing cycles in the third window of sequencing cycles, thereby substituting the respective compressed spatial output sets for the overlapping sequencing cycles for the respective per-cycle analyte channel sets for the overlapping sequencing cycles.

5. The artificial intelligence-based method of clause 1, wherein each per-cycle analyte channel set in the series depicts intensities registered in response to nucleotide incorporation in analytes at a corresponding sequencing cycle in the sequencing run.

6. The artificial intelligence-based method of clause 5, wherein the spatial network has a sequence of spatial convolution layers that separately processes each per-cycle analyte channel set in a particular window of per-cycle analyte channel sets in the series for a particular window of sequencing cycles of the sequencing run, and produce a sequence of spatial output sets for each sequencing cycle in the particular window of sequencing cycles, including beginning with a first spatial convolution layer that combines intensities only within a per-cycle analyte channel set of a subject sequencing cycle and not between per-cycle analyte channel sets of different sequencing cycles in the particular window of sequencing cycles, and continuing with successive spatial convolution layers that combine spatial outputs of preceding spatial convolution layers only within a subject sequencing cycle and not between the different sequencing cycles in the particular window of sequencing cycles.

7. The artificial intelligence-based method of clause 6, wherein respective spatial convolution layers in the sequence of spatial convolution layers have different counts of convolution filters, wherein a final spatial convolution layer in the sequence of spatial convolution layers has M convolution filters, and wherein M is an integer greater than four.

8. The artificial intelligence-based method of clause 7, wherein respective spatial convolution layers in the sequence of spatial convolution layers have a same count of convolution filters, wherein the same count is M, and wherein M is an integer greater than four.

9. The artificial intelligence-based method of clause 8, wherein the convolution filters in the spatial network use two-dimensional (2D) convolutions.

10. The artificial intelligence-based method of clause 8, wherein the convolution filters in the spatial network use three-dimensional (3D) convolutions.

11. The artificial intelligence-based method of clause 6, wherein the neural network-based base caller has a temporal network, wherein the temporal network has a sequence of temporal convolution layers that groupwise processes respective compressed spatial output sets for windows of successive sequencing cycles in the particular window of sequencing cycles, and produces a sequence of temporal output sets for the particular window of sequencing cycles, including beginning with a first temporal convolution layer that combines compressed spatial output sets between the different sequencing cycles in the particular window of sequencing cycles, and continuing with successive temporal convolution layers that combine successive temporal outputs of preceding temporal convolution layers.

12. The artificial intelligence-based method of clause 11, further including:

for the first window of sequencing cycles,

processing, through a first temporal convolution layer in the sequence of temporal convolution layers of the temporal network, the respective compressed spatial output sets for windows of successive sequencing cycles in the first window of sequencing cycles, and generating a plurality of temporal output sets for the first window of sequencing cycles;

processing, through the compression network, the plurality of temporal output sets, and generating respective compressed temporal output sets for respective temporal output sets in the plurality of temporal output sets, wherein the respective temporal output sets have M channels (feature maps), wherein the respective compressed temporal output sets have N channels (feature maps), and wherein M > N;

processing, through a final temporal convolution layer in the sequence of temporal convolution layers of the temporal network, the respective compressed temporal output sets, and generating a final temporal output set for the first window of sequencing cycles; and

generating, based on the final temporal output set, the base call predictions for one or more sequencing cycles in the first window of sequencing cycles,

wherein an output layer processes the final temporal output set and produces a final output for the first window of sequencing cycles, wherein the base call predictions are generated based on the final output.

13. The artificial intelligence-based method of clause 12, further including:

for the second window of sequencing cycles that shares, with the first window of sequencing cycles, one or more overlapping windows of successive sequencing cycles for which

the first temporal convolution layer previously generated temporal output sets, and at least one non-overlapping window of successive sequencing cycles for which the first temporal convolution layer is yet to generate a temporal output set,

processing, through the first temporal convolution layer, respective compressed spatial output sets only for respective sequencing cycles in the non-overlapping window of successive sequencing cycles, and generating a temporal output set for the non-overlapping window of successive sequencing cycles, thereby bypassing reprocessing through the first temporal convolution layer respective compressed spatial output sets for respective sequencing cycles in the overlapping windows of successive sequencing cycles;

processing, through the compression network, the temporal output set, and generating a compressed temporal output set for the non-overlapping window of successive sequencing cycles, wherein the temporal output set has M channels (feature maps), wherein the compressed temporal output has N channels (feature maps), and wherein M > N;

processing, through the final temporal convolution layer, respective compressed temporal output sets for the overlapping windows of successive sequencing cycles previously generated for the first window of sequencing cycles and on the compressed temporal output set, and generating a final temporal output set for the second window of sequencing cycles, thereby substituting the respective compressed temporal output sets for the overlapping windows of successive sequencing cycles for the respective per-cycle analyte channel sets for the overlapping windows of successive sequencing cycles; and

generating, based on the final temporal output set, the base call predictions for one or more sequencing cycles in the second window of sequencing cycles,

wherein an output layer processes the final temporal output set and produces a final output for the second window of sequencing cycles, wherein the base call predictions are generated based on the final output.

14. The artificial intelligence-based method of clause 13, further including:

for the third window of sequencing cycles that shares, with the first and second windows of sequencing cycles, one or more overlapping windows of successive sequencing cycles for which the first temporal convolution layer previously generated temporal output sets, and at least one non-overlapping window of successive sequencing cycles for which the first temporal convolution layer is yet to generate a temporal output set,

processing, through the first temporal convolution layer, respective compressed spatial output sets only for respective sequencing cycles in the non-overlapping window of successive sequencing cycles, and generating a temporal output set for the non-overlapping window of successive sequencing cycles, thereby bypassing reprocessing, through the first temporal convolution layer, respective compressed spatial output sets for respective sequencing cycles in the overlapping windows of successive sequencing cycles;

processing, through the compression network, the temporal output set, and generating a compressed temporal output set for the non-overlapping window of successive sequencing cycles, wherein the temporal output set has M channels (feature maps), wherein the compressed temporal output has N channels (feature maps), and wherein M > N; and

processing, through the final temporal convolution layer, respective compressed temporal output sets for the overlapping windows of successive sequencing cycles previously generated for the first and second windows of sequencing cycles and on the compressed temporal output set, and generating a final temporal output set for the third window of sequencing cycles, thereby substituting the respective compressed temporal output sets for the overlapping windows of successive sequencing cycles for the respective per-cycle analyte channel sets for the overlapping windows of successive sequencing cycles; and

generating, based on the final temporal output set, the base call predictions for one or more sequencing cycles in the third window of sequencing cycles,

wherein an output layer processes the final temporal output set and produces a final output for the third window of sequencing cycles, wherein the base call predictions are generated based on the final output.

15. The artificial intelligence-based method of clause 11, wherein respective temporal convolution layers in the sequence of temporal convolution layers of the temporal network have different counts of convolution filters, wherein the first temporal convolution layer has M convolution filters, and wherein M is an integer greater than four.

16. The artificial intelligence-based method of clause 11, wherein respective temporal convolution layers in the sequence of temporal convolution layers of the temporal network have a same count of convolution filters, wherein the same count is M, and wherein M is an integer greater than four.

17. The artificial intelligence-based method of clause 16, wherein the convolution filters in the temporal network use one-dimensional (1D) convolutions.

18. The artificial intelligence-based method of clause 1, wherein the compression network uses 1x1 convolutions to control a number of compressed spatial outputs in a compressed spatial output set, wherein the compression network has N convolution filters, and wherein N is an integer equal to or less than four.

19. The artificial intelligence-based method of clause 1, further including using data identifying unreliable analytes to remove portions of compressed spatial outputs in a compressed spatial output set corresponding to the unreliable analytes, and generating a compressed, filtered spatial output set to substitute the compressed spatial output set and generate base call predictions only for those analytes that are not the unreliable analytes.

20. The artificial intelligence-based method of clause 19, further including processing through the temporal network compressed, filtered spatial output sets instead of corresponding compressed spatial output sets.

21. The artificial intelligence-based method of clause 20, further including generating compressed temporal output sets from the compressed, filtered spatial output sets.

22. The artificial intelligence-based method of clause 19, wherein the data identifying the unreliable analytes identifies pixels that depict intensities of the unreliable clusters.

23. The artificial intelligence-based method of clause 19, wherein the data identifying the unreliable analytes identifies pixels that do not depict any intensities.

24. The artificial intelligence-based method of clause 20, wherein the compressed spatial output sets have four to nine times as many total pixels as the corresponding compressed, filtered spatial output sets.

25. The artificial intelligence-based method of clause 24, wherein the compressed, filtered spatial output sets cause the temporal network to operate on 75% fewer pixels, and thereby reduce compute operations, memory access, and memory occupancy for the temporal network by 75%.

26. The artificial intelligence-based method of clause 5, wherein bypassing reprocessing through the spatial network reduces compute operations, memory access, and memory occupancy for the temporal network by 80%.

27. The artificial intelligence-based method of clause 14, wherein bypassing reprocessing through the temporal network reduces compute operations, memory access, and memory occupancy for the temporal network.

28. The artificial intelligence-based method of clause 27, further including reallocating compute resources made available by the compression network to addition of supplemental convolution filters in the spatial network and the temporal network.

29. The artificial intelligence-based method of clause 27, further including reallocating compute resources made available by the compression network to addition of supplemental per-cycle analyte channel sets in each window of per-cycle analyte channel sets used to generate base call predictions for a particular sequence cycle.

30. The artificial intelligence-based method of clause 27, further including reallocating compute resources made available by the compression network to addition of supplemental spatial convolution layers in the spatial network.

31. The artificial intelligence-based method of clause 27, further including reallocating compute resources made available by the compression network to addition of supplemental temporal convolution layers in the temporal network.

32. The artificial intelligence-based method of clause 1, further including using one or more compressed spatial output sets generated for one or more preceding windows of sequence cycles in conjunction with one or more compressed spatial output sets generated for a current window of sequence cycles to generate base call predictions for one or more sequencing cycles in the current window of sequencing cycles.

33. The artificial intelligence-based method of clause 1, further including using one or more compressed spatial output sets generated for one or more succeeding windows of sequence cycles in conjunction with one or more compressed spatial output sets generated for a current window of sequence cycles to generate base call predictions for one or more sequencing cycles in the current window of sequencing cycles.

34. The artificial intelligence-based method of clause 1, further including using one or more compressed temporal output sets generated for one or more preceding windows of sequence cycles in conjunction with one or more compressed temporal output sets generated for a current window of sequence cycles to generate base call predictions for one or more sequencing cycles in the current window of sequencing cycles.

35. The artificial intelligence-based method of clause 1, further including using one or more compressed temporal output sets generated for one or more succeeding windows of sequence cycles in conjunction with one or more compressed temporal output sets generated for a current window of sequence cycles to generate base call predictions for one or more sequencing cycles in the current window of sequencing cycles.

36. The artificial intelligence-based method of clause 1, wherein a per-cycle analyte channel set encodes analyte data for analytes sequenced during the sequencing run.

37. The artificial intelligence-based method of clause 36, wherein the analyte data is image data that identifies intensity emissions collected from the analytes.

38. The artificial intelligence-based method of clause 37, wherein the image data has a plurality of image channels (images).

39. The artificial intelligence-based method of clause 38, wherein an image channel (image) is generated by a combination of (i) illumination with a specific laser and (ii) imaging through a specific optical filter.

40. The artificial intelligence-based method of clause 36, wherein the analyte data is electrical current and/or voltage data detected based on analyte activity.

41. The artificial intelligence-based method of clause 36, wherein the analyte data is pH scale data detected based on analyte activity.

42. The artificial intelligence-based method of clause 1, wherein a number of channels in each per-cycle analyte channel set in the series determines a number of convolution filters in the compression network and therefore a number of channels in compressed spatial output sets and compressed temporal output sets.

43. The artificial intelligence-based method of clause 1, wherein the compressed spatial output sets, the compressed, filter spatial output sets, and the compressed temporal output sets are stored in a quantized form.

44. A system, comprising:

host memory attached to a host processor and configured to receive a progression of sequencing images as a sequencing run progresses;

a configurable processor having an array of processing units, processing units in the array of processing units configured to execute a neural network-based base caller to produce base call predictions;

data flow logic having access to the host memory, the host processor, and the configurable processor, and configured to load sequencing images for sequencing cycles in a first window of sequencing cycles on the configurable processor from the host memory;

runtime logic configured to cause the processing units to execute a spatial network of the neural network-based base caller on the sequencing images for the sequencing cycles in the first window of sequencing cycles on a cycle-by-cycle basis and generate spatial feature map sets for each of the sequencing cycles in the first window of sequencing cycles;

the runtime logic configured to cause the processing units to execute a compression network of the neural network-based base caller on the spatial feature map sets on the cycle-by-cycle basis and generate compressed spatial feature map sets, and process the compressed spatial feature maps sets through a temporal network and an output network to produce base call predications for one or more sequencing cycles in the first window of sequencing cycles;

the data flow logic configured to move the compressed spatial feature map sets to the host memory and overwrite the sequencing images with the compressed spatial feature map sets;

for a second window of sequencing cycles that shares one or more overlapping sequencing cycles with the first window of sequencing cycles, and has at least one non-overlapping sequencing cycle, the data flow logic configured to load, on the configurable processor from the host memory, compressed spatial feature map sets for the overlapping sequencing cycles, and sequencing images for the non-overlapping sequencing cycle;

the runtime logic configured to cause the processing units to execute the spatial network on the sequencing images for the non-overlapping sequencing cycle and generate a spatial feature map set for the non-overlapping sequencing cycle; and

the runtime logic configured to cause the processing units to execute the compression network on the spatial feature map set and generate a compressed spatial feature map set for the non-overlapping sequencing cycle, and process the compressed spatial feature maps sets for the overlapping sequencing cycles and the compressed spatial feature map set for the non-overlapping sequencing cycle through the temporal network and the output network to produce base call predications for one or more sequencing cycles in the second window of sequencing cycles.

45. A system, comprising:

host memory attached to a host processor and configured to receive a progression of sequencing images as a sequencing run progresses;

a configurable processor having an array of processing units attached to processor memory, processing units in the array of processing units configured to execute a neural network-based base caller to produce base call predictions;

data flow logic having access to the host memory, the host processor, the configurable processor, and the processor memory, and configured to load sequencing images for sequencing cycles in a first window of sequencing cycles on the configurable processor from the host memory;

runtime logic configured to cause the processing units to execute a spatial network of the neural network-based base caller on the sequencing images for the sequencing cycles in the first window of sequencing cycles on a cycle-by-cycle basis and generate spatial feature map sets for each of the sequencing cycles in the first window of sequencing cycles;

the runtime logic configured to cause the processing units to execute a compression network of the neural network-based base caller on the spatial feature map sets on the cycle-by-cycle basis and generate compressed spatial feature map sets, and process the compressed spatial feature maps sets through a temporal network and an output network to produce base call predications for one or more sequencing cycles in the first window of sequencing cycles;

the data flow logic configured to move the compressed spatial feature map sets to the processor memory;

for a second window of sequencing cycles that shares one or more overlapping sequencing cycles with the first window of sequencing cycles, and has at least one non-overlapping sequencing cycle, the data flow logic configured to load, on the configurable processor from the processor memory, compressed spatial feature map sets for the overlapping sequencing cycles, and, load from the host memory, sequencing images for the non-overlapping sequencing cycle;

the runtime logic configured to cause the processing units to execute the spatial network on the sequencing images for the non-overlapping sequencing cycle and generate a spatial feature map set for the non-overlapping sequencing cycle; and

the runtime logic configured to cause the processing units to execute the compression network on the spatial feature map set and generate a compressed spatial feature map set for the non-overlapping sequencing cycle, and process the compressed spatial feature maps sets for the overlapping sequencing cycles and the compressed spatial feature map set for the non-overlapping sequencing cycle through the temporal network and the output network to produce base call predications for one or more sequencing cycles in the second window of sequencing cycles.

46. A system, comprising:

neural network logic configured to execute a first traversal of a neural network graph to independently process respective inputs in a first set of inputs through a first processing logic and generate respective alternative representations of the respective inputs in the first set of inputs without mixing information between the respective inputs in the first set of inputs, and produce outputs for the first traversal based on the respective alternative representations of the respective inputs in the first set of inputs;

the neural network logic configured to execute a second traversal of the neural network graph to independently process respective inputs in a second set of inputs through the first processing logic and generate respective alternative representations of the respective inputs in the second set of inputs without mixing information between the respective inputs in the second set of inputs, and produce outputs for the second traversal based on the respective alternative representations of the respective inputs in the second set of inputs, wherein the first and second set of inputs have one or more overlapping inputs and at least one non-overlapping input;

runtime logic configured with the neural network logic to execute the first traversal to generate the respective alternative representations of the respective inputs in the first set of inputs, to store the respective alternative representations of the respective inputs in the first set of inputs in memory in a compressed form, and to produce the outputs for the first traversal based on the compressed form of the respective alternative representations of the respective inputs in the first set of inputs; and

the runtime logic configured to execute the second traversal to process only the non-overlapping input through the first processing logic and generate an alternative representation of the non-overlapping input, to store the alternative representation of the non-overlapping input in memory in the compressed form, to retrieve the compressed form of respective alternative representations of the overlapping inputs generated in the first traversal to compensate for bypassing redundant generation of the respective alternative representations of the overlapping inputs in the second traversal, and to produce the outputs for the second traversal based on the compressed form of the respective alternative representations of the overlapping inputs and the compressed form of the alternative representation of the non-overlapping input.

47. The system of clause 46, wherein the memory is on-chip memory.

48. The system of clause 46, wherein the memory is off-chip memory.

49. The system of clause 46, wherein a number of channels in the compressed form corresponds to a number of channels in the inputs in the first and second set of inputs.

50. An artificial intelligence-based method of base calling, the method including:

accessing a series of per-cycle analyte channel sets generated for sequencing cycles of a sequencing run, wherein a subject per-cycle analyte channel set encodes analyte data detected for analytes at a subject sequencing cycle of the sequencing run;

processing the subject per-cycle analyte channel set through a first processing module of a neural network and producing an intermediate representation of the subject per-cycle analyte channel set with M feature maps;

processing the intermediate representation through a second processing module of the neural network and producing a reduced intermediate representation of the subject per-cycle analyte channel set with N feature maps, where M > N; and

using the reduced intermediate representation of the subject per-cycle analyte channel set to generate base call predictions for the analytes at the subject sequencing cycle and/or at other sequencing cycles of the sequencing run.

51. The artificial intelligence-based method of clause 50, wherein the first processing module is a convolution layer with M convolution filters.

52. The artificial intelligence-based method of clause 50, wherein the second processing module is a convolution layer with N convolution filters.

53. An artificial intelligence-based method of base calling, the method including:

processing a progression of per-cycle analyte channel sets generated for sequencing cycles of a sequencing run through a neural network-based base caller on a sliding window basis such that successive sliding windows have overlapping sequencing cycles, including:

for a current window of the sequencing cycles that comprises one or more preceding sequencing cycles, a center sequencing cycle, and one or more succeeding sequencing cycles:

generating a spatial intermediate representation and a compressed intermediate representation for each of the preceding sequencing cycles, the center sequencing cycle, and the succeeding sequencing cycles based on applying the neural network-based base caller on the current window of the per-cycle analyte channel sets, wherein the spatial intermediate representation has M channels, the compressed intermediate representation has N channels, and M > N; and

base calling at least the center sequencing cycle based on the compressed intermediate representations generated for the preceding sequencing cycles, the center sequencing cycle, and the succeeding sequencing cycles; and

using the compressed intermediate representations generated for the preceding sequencing cycles, the center sequencing cycle, and the succeeding sequencing cycles to base call at least at a center sequencing cycle in a next window of the sequencing cycles.

54. An artificial intelligence-based system for base calling, the system comprising:

a host processor;

memory accessible by the host processor storing analyte data for sequencing cycles of a sequencing run; and

a configurable processor having access to the memory, the configurable processor including:

a plurality of execution clusters, the execution clusters in the plurality of execution clusters configured to execute a neural network; and

data flow logic having access to the memory and to the execution clusters in the plurality of execution clusters, configured to provide the analyte data to available execution clusters in the plurality of execution clusters, cause the execution clusters to apply the analyte data to the neural network to generate intermediate representations and compressed intermediate representations of the analyte data for use in a current base calling step, and to feed back the compressed intermediate representations to the memory for use in a future base calling step as a replacement of the analyte data, wherein an intermediate representation as M channels, a compressed intermediate representation has N channels, and M > N.

55. A system, comprising:

runtime logic configured to execute a first iteration of a base caller to process an input and generate intermediate representations of the input;

compression logic configured to process the intermediate representations and generate compressed intermediate representations of the input; and

the runtime logic configured to use the compressed intermediate representations in lieu of the input in a subsequent iteration of the base caller.

56. A system, comprising:

runtime logic configured to execute a first iteration of a base caller to process an input and generate intermediate representations of the input;

compression logic configured to process the intermediate representations and generate compressed intermediate representations, wherein the compressed intermediate representations are configured to have as many channels as the input; and

the runtime logic configured to use the compressed intermediate representations in lieu of the input in a subsequent iteration of the base caller.

57. The system of clause 56, wherein the channels correspond to feature maps.

58. The system of clause 56, wherein the channels correspond to a depth dimension.

59. The system of clause 56, wherein the channels correspond to spatial dimensions.

Split

1. A system, comprising:

a spatial convolution network configured to process a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately convolving respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective sequences of spatial convolution layers to generate respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles;

wherein the respective sequences of spatial convolution layers have respective sequences of spatial convolution filter banks, wherein trained coefficients of spatial convolution filters in spatial convolution filter banks of the respective sequences of spatial convolution filter banks vary between sequences of spatial convolution layers in the respective sequences of spatial convolution layers;

a temporal convolution network configured to process the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks of a first temporal convolution layer to generate respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets; and

wherein trained coefficients of temporal convolution filters in the respective temporal convolution filter banks vary between temporal convolution filter banks in the respective temporal convolution filter banks.

2. The system of clause 1, wherein the spatial convolution filters use intra-cycle segregated convolutions.

3. The system of clause 1, wherein the temporal convolution filters use inter-cycle combinatory convolutions.

4. The system of clause 1, further configured to comprise a compression network that separately convolves the respective per-cycle spatial feature map sets through respective compression convolution layers to generate respective per-cycle compressed spatial feature map sets for the respective sequencing cycles.

5. The system of clause 4, wherein trained coefficients of compression convolution filters in the respective compression convolution layers vary between compression convolution layers in the respective compression convolution layers.

6. The system of clause 5, wherein the temporal convolution network is further configured to process the per-group temporal feature map sets on the groupwise basis by convolving on respective overlapping groups of per-group temporal feature map sets in the per-group temporal feature map sets using respective temporal convolution filter banks of a second temporal convolution layer to generate respective further per-group temporal feature map sets for the respective overlapping groups of per-group temporal feature map sets.

7. The system of clause 6, further configured to comprise an output network that processes a final temporal feature map set generated by a final temporal convolution layer to generate a final output.

8. The system of clause 7, further configured to produce base call predications for one or more sequencing cycles in the series of sequencing cycles based on the final output.

9. A system, comprising:

a spatial convolution network configured to process a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately convolving respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective sequences of spatial convolution layers to generate respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles;

a temporal convolution network configured to process the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks to generate respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets; and

wherein trained coefficients of temporal convolution filters in the respective temporal convolution filter banks vary between temporal convolution filter banks in the respective temporal convolution filter banks.

10. The system of clause 9, wherein the respective sequences of spatial convolution layers have respective sequences of spatial convolution filter banks, wherein trained coefficients of spatial convolution filters in spatial convolution filter banks of the respective sequences of spatial convolution filter banks are shared between sequences of spatial convolution layers in the respective sequences of spatial convolution layers.

11. The system of clause 9, further configured to comprise a compression network that separately convolves the respective per-cycle spatial feature map sets through respective compression convolution layers to generate respective per-cycle compressed spatial feature map sets for the respective sequencing cycles, wherein trained coefficients of compression convolution filters in the respective compression convolution layers vary between compression convolution layers in the respective compression convolution layers.

12. An artificial intelligence-based method of base calling, the method including:

processing, through spatial convolution network, a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately convolving respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective sequences of spatial convolution layers, and generating respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles;

wherein the respective sequences of spatial convolution layers have respective sequences of spatial convolution filter banks, wherein trained coefficients of spatial convolution filters in spatial convolution filter banks of the respective sequences of spatial convolution filter banks vary between sequences of spatial convolution layers in the respective sequences of spatial convolution layers;

processing, through a temporal convolution network, the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks of a first temporal convolution layer, and generating respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets; and

wherein trained coefficients of temporal convolution filters in the respective temporal convolution filter banks vary between temporal convolution filter banks in the respective temporal convolution filter banks.

13. The artificial intelligence-based method of clause 12, further including separately convolving the respective per-cycle spatial feature map sets through respective compression convolution layers of a compression network and generating respective per-cycle compressed spatial feature map sets for the respective sequencing cycles.

14. The artificial intelligence-based method of clause 13, wherein trained coefficients of compression convolution filters in the respective compression convolution layers vary between compression convolution layers in the respective compression convolution layers.

15. The artificial intelligence-based method of clause 14, further including processing, through the temporal convolution network, the per-group temporal feature map sets on the groupwise basis by convolving on respective overlapping groups of per-group temporal feature map sets in the per-group temporal feature map sets using respective temporal convolution filter banks of a second temporal convolution layer, and generating respective further per-group temporal feature map sets for the respective overlapping groups of per-group temporal feature map sets.

16. The artificial intelligence-based method of clause 15, further including processing, through an output network, a final temporal feature map set generated by a final temporal convolution layer, and generating a final output.

17. The artificial intelligence-based method of clause 16, further including producing base call predications for one or more sequencing cycles in the series of sequencing cycles based on the final output.

18. An artificial intelligence-based method of base calling, the method including:

processing, through spatial convolution network, a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately convolving respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective sequences of spatial convolution layers, and generating respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles;

processing, through a temporal convolution network, the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter

banks of a first temporal convolution layer, and generating respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets; and

wherein trained coefficients of temporal convolution filters in the respective temporal convolution filter banks vary between temporal convolution filter banks in the respective temporal convolution filter banks.

19. The artificial intelligence-based method of clause 18, wherein the respective sequences of spatial convolution layers have respective sequences of spatial convolution filter banks, wherein trained coefficients of spatial convolution filters in spatial convolution filter banks of the respective sequences of spatial convolution filter banks are shared between sequences of spatial convolution layers in the respective sequences of spatial convolution layers.

20. The artificial intelligence-based method of clause 18, further including separately convolving the respective per-cycle spatial feature map sets through respective compression convolution layers of a compression network and generating respective per-cycle compressed spatial feature map sets for the respective sequencing cycles, wherein trained coefficients of compression convolution filters in the respective compression convolution layers vary between compression convolution layers in the respective compression convolution layers.

21. A system, comprising:

a spatial convolution network configured to apply respective sequences of spatial convolution layers to respective per-cycle sequencing images in a window of per-cycle sequencing images; and

wherein the respective sequences of spatial convolution layers have respective sequences of spatial convolution filter banks that differ from one sequence of spatial convolution layers to another sequence of spatial convolution layers.

22. A system, comprising:

a temporal convolution network configured with a first temporal convolution layer configured to apply respective sets of temporal convolution filters to respective sliding windows of spatial feature maps; and

wherein the respective sets of temporal convolution filters in the first temporal convolution layer have temporal convolution filters that differ from one set of temporal convolution filters to another set of temporal convolution filters.

23. The system of clause 22, wherein the temporal convolution network is configured with a second temporal convolution layer that succeeds the first temporal convolution layer, wherein

the second convolution layer is configured to apply respective sets of temporal convolution filters to respective sliding windows of temporal feature maps, and wherein the respective sets of temporal convolution filters in the second temporal convolution layer have temporal convolution filters that differ from one set of temporal convolution filters to another set of temporal convolution filters.

Skip

1. A system, comprising:

a spatial convolution network configured to process a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately processing respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective spatial processing pipelines, the respective spatial processing pipelines configured to convolve the respective per-cycle sequencing image sets through respective sequences of spatial convolution layers to generate respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles; and

a bus network, connected to the spatial convolution network, and configured to form buses between spatial convolution layers within the respective sequences of spatial convolution layers, the buses configured to cause respective per-cycle spatial feature map sets generated by two or more spatial convolution layers in a particular sequence of spatial convolution layer for a particular sequencing cycle to combine into a combined per-cycle spatial feature map set, and provide the combined per-cycle spatial feature map set as input to another spatial convolution layer in the particular sequence of spatial convolution layer.

2. The system of clause 1, wherein the two or more spatial convolution layers include a first spatial convolution layer and a third spatial convolution layer, wherein the first spatial convolution layer generates a first per-cycle spatial feature map set, wherein the first spatial convolution layer provides the first per-cycle spatial feature map set as input to a second spatial convolution layer, wherein the second spatial convolution layer processes the first per-cycle spatial feature map set and generates a second per-cycle spatial feature map set, wherein the second spatial convolution layer provides the second per-cycle spatial feature map set as input to the third spatial convolution layer, and wherein the third spatial convolution layer processes the second per-cycle spatial feature map set and generates a third per-cycle spatial feature map set.

3. The system of clause 2, wherein the buses are further configured to cause the first spatial feature map set and the third per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

4. The system of clause 3, wherein the another spatial convolution layer is a fourth spatial convolution layer that immediately succeeds the third spatial convolution layer in the particular sequence of spatial convolution layer, wherein the fourth spatial convolution layer processes the combined per-cycle spatial feature map set as input.

5. The system of clause 2, wherein the two or more spatial convolution layers include the first spatial convolution layer and a seventh spatial convolution layer, wherein the third spatial convolution layer provides the third per-cycle spatial feature map set as input to a fourth spatial convolution layer, wherein the fourth spatial convolution layer processes the third per-cycle spatial feature map set and generates a fourth per-cycle spatial feature map set, wherein the fourth spatial convolution layer provides the fourth per-cycle spatial feature map set as input to a fifth spatial convolution layer, and wherein the fifth spatial convolution layer processes the fourth per-cycle spatial feature map set and generates a fifth per-cycle spatial feature map set.

6. The system of clause 5, wherein the buses are further configured to cause the first spatial feature map set and the fifth spatial feature map set to combine into the combined per-cycle spatial feature map set.

7. The system of clause 6, wherein the another spatial convolution layer is a sixth spatial convolution layer that immediately succeeds the fifth spatial convolution layer in the particular sequence of spatial convolution layer, wherein the sixth spatial convolution layer processes the combined per-cycle spatial feature map set as input.

8. The system of clause 5, wherein the two or more spatial convolution layers include the first spatial convolution layer, the third spatial convolution layer, and the fifth spatial convolution layer, and wherein the buses are further configured to cause the first per-cycle spatial feature map set, the third per-cycle spatial feature map set, and the fifth per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

9. The system of clause 8, wherein the another spatial convolution layer is the sixth spatial convolution layer that processes the combined per-cycle spatial feature map set as input.

10. The system of clause 1, wherein the buses are further configured to cause a per-cycle sequencing image set for the particular sequencing cycle, provided as input to the first spatial convolution layer, and the third per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

11. The system of clause 10, wherein the another spatial convolution layer is the fourth spatial convolution layer that processes the combined per-cycle spatial feature map set as input.

12. The system of clause 1, wherein the bus network is further configured to include dimensionality compatibility logic configured to modify spatial and depth dimensionality of an incoming per-cycle spatial feature map set that is combined with a receiving per-cycle spatial feature map set to generate the combined per-cycle spatial feature map set.

13. The system of clause 12, wherein the dimensionality compatibility logic is a dimensionality reduction operation, including convolution, pooling, or averaging.

14. The system of clause 12, wherein the bus network is further configured to include scaling logic configured to scale feature values of the incoming per-cycle spatial feature map set that is combined with the receiving per-cycle spatial feature map set to generate the combined per-cycle spatial feature map set.

15. The system of clause 1, further configured to comprise a temporal convolution network configured to process the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks of a first temporal convolution layer to generate respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets.

16. The system of clause 15, further configured to comprise the bus network, connected to the temporal convolution network, and configured to form buses between temporal convolution layers within the respective sequences of temporal convolution layers, the buses configured to cause respective per-cycle temporal feature map sets generated by two or more temporal convolution layers in a particular sequence of temporal convolution layer for a particular sequencing cycle to combine into a combined per-cycle temporal feature map set, and provide the combined per-cycle temporal feature map set as input to another temporal convolution layer in the particular sequence of temporal convolution layer.

17. An artificial intelligence-based method, including:

processing, through a spatial convolution network, a window of per-cycle sequencing image sets for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately processing respective per-cycle sequencing image sets in the window of per-cycle sequencing image sets through respective spatial processing pipelines, including convolving the respective per-cycle sequencing image sets through respective sequences of spatial convolution layers to

generate respective per-cycle spatial feature map sets for respective sequencing cycles in the series of sequencing cycles; and

combining respective per-cycle spatial feature map sets generated by two or more spatial convolution layers in a particular sequence of spatial convolution layer for a particular sequencing cycle into a combined per-cycle spatial feature map set, and providing the combined per-cycle spatial feature map set as input to another spatial convolution layer in the particular sequence of spatial convolution layer.

18. The artificial intelligence-based method of clause 17, wherein the two or more spatial convolution layers include a first spatial convolution layer and a third spatial convolution layer, wherein the first spatial convolution layer generates a first per-cycle spatial feature map set, wherein the first spatial convolution layer provides the first per-cycle spatial feature map set as input to a second spatial convolution layer, wherein the second spatial convolution layer processes the first per-cycle spatial feature map set and generates a second per-cycle spatial feature map set, wherein the second spatial convolution layer provides the second per-cycle spatial feature map set as input to the third spatial convolution layer, and wherein the third spatial convolution layer processes the second per-cycle spatial feature map set and generates a third per-cycle spatial feature map set.

19. The artificial intelligence-based method of clause 18, wherein the buses are further configured to cause the first spatial feature map set and the third per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

20. The artificial intelligence-based method of clause 19, wherein the another spatial convolution layer is a fourth spatial convolution layer that immediately succeeds the third spatial convolution layer in the particular sequence of spatial convolution layer, wherein the fourth spatial convolution layer processes the combined per-cycle spatial feature map set as input.

[0308] Other implementations of the method described above 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 memory and one or more processors operable to execute instructions, stored in the memory, to perform any of the methods described above.

[0309] What is claimed is:

CLAIMS

1. A system, comprising:

a spatial convolution network configured to process a window of per-cycle sequencing image sets depicting clusters for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately processing respective per-cycle sequencing image sets depicting clusters in the window of per-cycle sequencing image sets through respective spatial processing pipelines;

wherein the image sets depicting clusters depict intensity emissions generated as a result of nucleotide incorporation in clusters of associated analytes on a substrate during a sequencing cycle of a sequencing run; and

a bus network, connected to the spatial convolution network, and configured to form buses between spatial convolution layers within respective sequences of spatial convolution layers of the respective spatial processing pipelines, the buses configured to cause respective per-cycle spatial feature map sets generated by two or more spatial convolution layers in a particular sequence of spatial convolution layers for a particular sequencing cycle to combine into a combined per-cycle spatial feature map set, and provide the combined per-cycle spatial feature map set as input to another spatial convolution layer in the particular sequence of spatial convolution layers.

2. The system of claim 1, wherein the two or more spatial convolution layers include a first spatial convolution layer and a third spatial convolution layer, wherein the first spatial convolution layer generates a first per-cycle spatial feature map set, wherein the first spatial convolution layer provides the first per-cycle spatial feature map set as input to a second spatial convolution layer, wherein the second spatial convolution layer processes the first per-cycle spatial feature map set and generates a second per-cycle spatial feature map set, wherein the second spatial convolution layer provides the second per-cycle spatial feature map set as input to the third spatial convolution layer, and wherein the third spatial convolution layer processes the second per-cycle spatial feature map set and generates a third per-cycle spatial feature map set.

3. The system of claim 2, wherein the buses are further configured to cause the first per-cycle spatial feature map set and the third per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

4. The system of claim 3, wherein the another spatial convolution layer is a fourth spatial convolution layer that immediately succeeds the third spatial convolution layer in the particular sequence of spatial convolution layers, wherein the fourth spatial convolution layer processes the combined per-cycle spatial feature map set as input.

5. The system of claim 2, wherein the two or more spatial convolution layers include the first spatial convolution layer and a seventh spatial convolution layer, wherein the third spatial convolution layer provides the third per-cycle spatial feature map set as input to a fourth spatial convolution layer, wherein the fourth spatial convolution layer processes the third per-cycle spatial feature map set and generates a fourth per-cycle spatial feature map set, wherein the fourth spatial convolution layer provides the fourth per-cycle spatial feature map set as input to a fifth spatial convolution layer, and wherein the fifth spatial convolution layer processes the fourth per-cycle spatial feature map set and generates a fifth per-cycle spatial feature map set.

6. The system of claim 5, wherein the buses are further configured to cause the first per-cycle spatial feature map set and the fifth per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

7. The system of claim 6, wherein the another spatial convolution layer is a sixth spatial convolution layer that immediately succeeds the fifth spatial convolution layer in the particular sequence of spatial convolution layers, wherein the sixth spatial convolution layer processes the combined per-cycle spatial feature map set as input.

8. The system of claim 5, wherein the two or more spatial convolution layers include the first spatial convolution layer, the third spatial convolution layer, and the fifth spatial convolution layer, and wherein the buses are further configured to cause the first per-cycle spatial feature map set, the third per-cycle spatial feature map set, and the fifth per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

9. The system claim 8, wherein the another spatial convolution layer is a sixth spatial convolution layer that processes the combined per-cycle spatial feature map set as input.

10. The system of claim 1, wherein the buses are further configured to cause a per-cycle sequencing image set for the particular sequencing cycle, provided as input to the first spatial

convolution layer, and the third per-cycle spatial feature map set to combine into the combined per-cycle spatial feature map set.

11. The system of claim 10, wherein the another spatial convolution layer is a fourth spatial convolution layer that processes the combined per-cycle spatial feature map set as input.

12. The system of claim 1, wherein the bus network is further configured to include dimensionality compatibility logic configured to modify spatial and depth dimensionality of an incoming per-cycle spatial feature map set that is combined with a receiving per-cycle spatial feature map set to generate the combined per-cycle spatial feature map set.

13. The system of claim 12, wherein the dimensionality compatibility logic is a dimensionality reduction operation, including convolution, pooling, or averaging.

14. The system of claim 12, wherein the bus network is further configured to include a scaling logic configured to scale feature values of the incoming per-cycle spatial feature map set that is combined with the receiving per-cycle spatial feature map set to generate the combined per-cycle spatial feature map set.

15. The system of claim 1, further configured to comprise a temporal convolution network configured to process the per-cycle spatial feature map sets on a groupwise basis by convolving on respective overlapping groups of per-cycle spatial feature map sets in the per-cycle spatial feature map sets using respective temporal convolution filter banks of a first temporal convolution layer to generate respective per-group temporal feature map sets for the respective overlapping groups of per-cycle spatial feature map sets.

16. The system of claim 15, further configured to comprise the bus network, connected to the temporal convolution network, and configured to form buses between temporal convolution layers within respective sequences of temporal convolution layers, the buses configured to cause respective per-cycle temporal feature map sets generated by two or more temporal convolution layers in a particular sequence of temporal convolution layers for a particular sequencing cycle to combine into a combined per-cycle temporal feature map set, and provide the combined per-cycle temporal feature map set as input to another temporal convolution layer in the particular sequence of temporal convolution layers.

17. An artificial intelligence-based method, including:

processing, through a spatial convolution network, a window of per-cycle sequencing image sets depicting clusters for a series of sequencing cycles of a sequencing run on a cycle-by-cycle basis by separately processing respective per-cycle sequencing image sets depicting clusters in the window of per-cycle sequencing image sets through respective sequences of spatial convolution layers;

wherein the image sets depicting clusters depict intensity emissions generated as a result of nucleotide incorporation in clusters of associated analytes on a substrate during a sequencing cycle of a sequencing run; and

combining respective per-cycle spatial feature map sets generated by two or more spatial convolution layers in a particular sequence of spatial convolution layers for a particular sequencing cycle into a combined per-cycle spatial feature map set, and providing the combined per-cycle spatial feature map set as input to another spatial convolution layer in the particular sequence of spatial convolution layers.

18. The artificial intelligence-based method of claim 17, wherein the two or more spatial convolution layers include a first spatial convolution layer and a third spatial convolution layer, wherein the first spatial convolution layer generates a first per-cycle spatial feature map set, wherein the first spatial convolution layer provides the first per-cycle spatial feature map set as input to a second spatial convolution layer, wherein the second spatial convolution layer processes the first per-cycle spatial feature map set and generates a second per-cycle spatial feature map set, wherein the second spatial convolution layer provides the second per-cycle spatial feature map set as input to the third spatial convolution layer, and wherein the third spatial convolution layer processes the second per-cycle spatial feature map set and generates a third per-cycle spatial feature map set.

19. The artificial intelligence-based method of claim 18, further comprising combining the first per-cycle spatial feature map set and the third per-cycle spatial feature map set into the combined per-cycle spatial feature map set.

20. The artificial intelligence-based method of claim 19, wherein the another spatial convolution layer is a fourth spatial convolution layer that immediately succeeds the third spatial convolution layer in the particular sequence of spatial convolution layers, wherein the fourth spatial convolution layer processes the combined per-cycle spatial feature map set as input.