



(51) International Patent Classification:

H04N 19/164 (2014.01) H04N 19/61 (2014.01)

H04N 19/105 (2014.01) H04N 19/89 (2014.01)

H04N 19/172 (2014.01)

(21) International Application Number:

PCT/US2022/036476

(22) International Filing Date:

08 July 2022 (08.07.2022)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

17/365,669 30 July 2021 (30.07.2021) US

(71) Applicant: NVIDIA CORPORATION [US/US]; 2788

San Tomas Expressway, Santa Clara, California 95051 (US).

(72) Inventors: WAINE, Benjemin Thomas; 2788 San Tomas

Expressway, Santa Clara, California 95051 (US). ROSE, Amy; 2788 San Tomas Expressway, Santa Clara, Califor-

nia 95051 (US). WOODARD, Andrew James; 2788 San Tomas Expressway, Santa Clara, California 95051 (US).

(74) Agent: LOHR, Jason et al.; Hogan Lovells US LLP, 3 Embarcadero Center, Suite 1500, San Francisco, California 94111 (US).

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

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

(54) Title: VIDEO COMPRESSION TECHNIQUES FOR RELIABLE TRANSMISSION

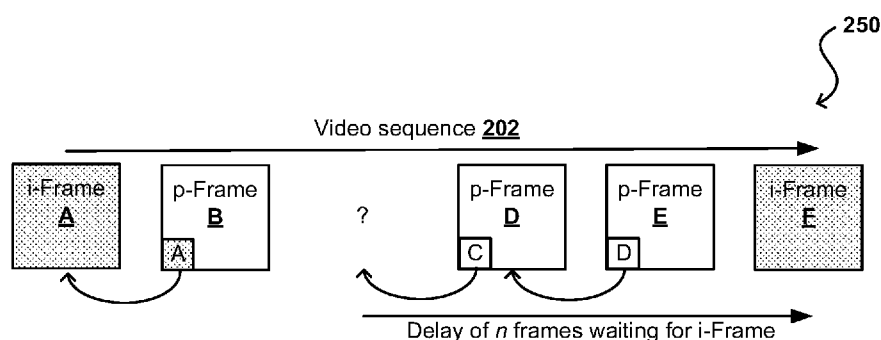


FIG. 2B

(57) Abstract: Systems and methods are presented for reliable transmission of time-sensitive data. In particular, various embodiments provide for the generation of compressed sequential data, where individual instances of a sequence represent differentials from prior instances in that sequence. In order to reduce an amount of data that needs to be transmitted, instances of data (such as individual video frames) can be provided using a prior video frame as a reference, sending only data for those pixel locations where the pixel value differs from the reference frame. A reference frame can include a previously-received and successfully-decoded frame, in order to minimize the impact of dropped, incomplete, or corrupted frames. In order to further reduce data transmission requirements, a reference frame can be selected which is determined to be optimal for the current frame, such as may represent a least amount of data to be transmitted for a given frame.

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

Declarations under Rule 4.17:

- *as to the identity of the inventor (Rule 4.17(i))*
- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
- *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

VIDEO COMPRESSION TECHNIQUES FOR RELIABLE TRANSMISSION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Non-Provisional Patent Application Serial No. 17/365,669, filed July 30, 2021, and entitled “VIDEO COMPRESSION TECHNIQUES FOR RELIABLE TRANSMISSION” which is hereby incorporated herein in its entirety and for all purposes.

BACKGROUND

[0002] Data streaming is becoming increasingly important for a variety of different industries and applications. For certain applications, such as online gaming, it can be important for the streaming of game or video data to have low latency, in order to avoid the latency negatively impacting gameplay. Using conventional video compression and streaming technology, data for video frames in a sequence may all reference changes with respect to an immediately prior frame in that sequence, other than for key frames (*e.g.*, “i-Frames”) that do not depend upon other frames in the sequence. This reliance on the prior frame can create issues when a frame is dropped, for example, as subsequent frames in that sequence will then be missing some of the change information needed to decode a proper frame. While another key frame can be sent to correct for this frame drop, there will be some latency experienced in generating a new key frame, as well as extra time and bandwidth needed to transmit another relatively large key frame.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] Various embodiments in accordance with the present disclosure will be described with reference to the drawings, in which:

[0004] FIGS. 1A, 1B, and 1C illustrate image data for an example image sequence, according to at least one embodiment;

[0005] FIGS. 2A and 2B illustrate a flow of frames in a first video sequence;

[0006] FIGS. 3A, 3B, 3C, and 3D illustrate a slow of frames in a second video sequence, according to at least one embodiment;

[0007] FIG. 4 illustrates a video encoding and transmission pipeline, according to at least one embodiment;

[0008] FIGS. 5A and 5B illustrate a process for dynamically determining a reference frame to compute one or more subsequent frame differentials, according to at least one embodiment;

[0009] FIG. 6 illustrates components of a system for encoding and transmitting image data, according to at least one embodiment;

[0010] FIG. 7A illustrates inference and/or training logic, according to at least one embodiment;

[0011] FIG. 7B illustrates inference and/or training logic, according to at least one embodiment;

[0012] FIG. 8 illustrates an example data center system, according to at least one embodiment;

[0013] FIG. 9 illustrates a computer system, according to at least one embodiment;

[0014] FIG. 10 illustrates a computer system, according to at least one embodiment;

[0015] FIG. 11 illustrates at least portions of a graphics processor, according to one or more embodiments;

[0016] FIG. 12 illustrates at least portions of a graphics processor, according to one or more embodiments;

[0017] FIG. 13 is an example data flow diagram for an advanced computing pipeline, in accordance with at least one embodiment;

[0018] FIG. 14 is a system diagram for an example system for training, adapting, instantiating and deploying machine learning models in an advanced computing pipeline, in accordance with at least one embodiment; and

[0019] FIGS. 15A and 15B illustrate a data flow diagram for a process to train a machine learning model, as well as client-server architecture to enhance annotation tools with pre-trained annotation models, in accordance with at least one embodiment.

DETAILED DESCRIPTION

[0020] Approaches in accordance with various embodiments can provide for the generation, compression, and/or encoding of sequential or differential data. In particular, various embodiments provide for the generation of compressed sequential data, where individual instances of a sequence may represent differentials from one or more prior instances in that sequence. This may include, for example and without limitation, frames in a video sequence. For at least some instances or applications, latency may be important or even critical. In order to reduce an amount of data that needs to be transmitted for such an application, the data can be compressed using any of a variety of compression techniques. One such technique is to send an initial video frame (or other data instance) that includes all relevant data for that frame, such as pixel values for each pixel location in that video frame. Subsequent frames can then be provided using the initial video frame as a reference, or reference frame, sending only data for those pixel locations where the pixel value differs from the reference frame. In various embodiments, a reference frame can correspond to a frame that was confirmed to have been previously received and successfully decoded, rather than an immediately prior frame in the sequence, in order to minimize the impact of dropped, incomplete, or corrupted frames. In order to further reduce data transmission requirements, a differential can be calculated with respect to one of a number of previously-verified or confirmed frames that may be cached on, for example, a local device. In at least one embodiment, differentials can be calculated or estimated for a current frame with respect to each of a number of potential reference frames. A reference frame can then be selected that is determined to be “optimal” for the current frame, such as may represent a least amount of differential data to be transmitted or a least amount of processing to decode or render. The set of possible reference frames can be updated as new frames are successfully verified or confirmed, such as where an acknowledgement message is received from a recipient device.

[0021] In at least one embodiment, an image generation system can generate such a sequence of images, or video frames, that when played in sequence will present a scene of content, such as a scene of animation or gameplay. In each individual image, there may be various objects represented, such as may include both foreground and background objects that may be static or dynamic. The location, size, and orientation of any of those objects may change, whether static or dynamic, based at least in part upon motion of a virtual camera that is used to determine factors, such as a point of view or zoom level, for the image. In such image or video sequences, these changes in view, location, size, and orientation can be viewed as a set of changes of individual pixels used to represent those objects. These changes between frames can be viewed as differentials in pixel data in at least some embodiments.

[0022] For online or real time applications or use cases, these video sequences can be delivered using techniques such as video streaming. In applications such as streaming video for movies, any delays or latency may be annoying to a user, but may not substantially prevent that user from viewing and enjoying the video. As mentioned, however, for other types of video streaming any significant latency may reduce enjoyment of the experience or even create an unacceptable experience. For example, the streaming of live sporting events may be at least somewhat latency-sensitive, as viewers may not want to miss, or have a delay in the viewing of, certain events in the live stream resulting from, for example, problems with the video feed. For applications such as video conferencing, significant delays may reduce the value or impact of the discussion. For applications such as online gaming, any significant latency issues can be significant, as it may result in the player losing or being at a significant disadvantage if the player is effectively unable to determine what is going on in the game for an extended period of time. For applications such as robotics or autonomous driving, where physical determinations might be made based at least in part upon the live video stream, any latency that results in erroneous actions being taken, or desired actions not being taken, can have dire consequences. For these and other such applications or use cases, it may not be sufficient to simply buffer the data that is received and deal with frame loss or corruption by waiting for frames to catch up or using a reliable transport. Any increase in latency resulting from such approaches can negatively impact the application, such as to make online games less interactive or automation applications slower

and less reactive. It can be desirable for at least some of these applications and use cases to be able to limit the impact of data loss.

[0023] As an example, consider the initial image frame 100 of FIG. 1A, which may correspond to a frame of a video sequence. For simplicity of explanation, this initial frame shows a solid black object 102 in front of a solid white background, in a low resolution image that includes an array of 18 x 18 pixels. In order to transmit this initial image, pixel data (i.e., the color to be displayed at each pixel location) must be transmitted for all 324 pixels. FIG. 1A also illustrates a subsequent frame 110 in that sequence, which shows the same black object 102 at a slightly different location. Without image compression, it would again be necessary to transmit pixel data for all 324 pixels in this subsequent frame, in order to properly render and display this subsequent frame 110.

[0024] In order to reduce the amount of data to be transmitted, one or more data compression techniques can be used that include only changes with respect to an immediately prior frame in the sequence. For example, consider the overlaid frames 120 of FIG. 1B. A comparison can be made between the initial frame 100 and the subsequent frame 110 that shows that many of the white background pixels remain white, and some of the pixels 124 corresponding to the object remain black. There are two regions of pixels where the values have changed. There is a first region 122 where the pixels were black in the initial image but changed to white background pixels in the subsequent image 110. There is a second region 126 where the white background pixels change to black pixels to represent the current location of the object. In order to reduce an amount of data to be sent for the subsequent image 110, data can be sent that represents only those pixels with changes 130 relative to the initial image 100. In other words, data for the 46 pixels with different values in the subsequent frame can be transmitted for the subsequent frame, while pixel data for the remaining 278 pixels can be retained and used from the initial frame 100. Even in this simple example, which ignores variations in color or aspects such as anti-aliasing, significant reduction in data transfer can be obtained.

[0025] In order for such a scheme to work without error, however, each frame in a sequence may need to be fully received and accurately decoded, as it serves as the basis for a subsequent

frame in the sequence. If a frame is dropped, this can result in a loss of data that can impact the visual quality of the video sequence, or can prevent frames from being properly decoded. As an example, consider the video sequence of FIG. 2A and FIG. 2B. The frames 200 of FIG. 2A belong to an example video sequence 202. An initial frame in this sequence is an i-Frame (A), which includes data for all pixels in that frame. As mentioned, an i-Frame (or other type of key frame) does not depend upon any prior frames in a sequence, and can stand alone as a full video frame or image. A subsequent frame (B), here a p-Frame (or a “progressive” frame that is representative of progress from the immediately prior frame) although any other relevant type of dependent frame or image data expression can be used as well, can depend on, or refer back to, i-Frame A, such that the data transmitted for p-Frame B includes only data for pixels whose values have changed (or changed with at least a minimum amount of difference) with respect to the values in i-Frame A. The small box in the lower left corner of p-Frame B indicates that the frame refers back to i-Frame A, and includes data for differences with respect to i-Frame A. As illustrated in this example sequence 202, each subsequent frame refers back to the immediately prior frame, and thus includes only changes with respect to the immediately prior frame.

[0026] It might be the case, however, that data (*e.g.*, one or more packets) for one or more of the frames in video sequence 202 may be dropped, partially received, corrupted, or otherwise unable to be used to decode a proper image or video frame. As illustrated for the frames 250 received in the video sequence 202 of FIG. 2B, p-Frame C was not completely and accurately received. As illustrated, p-Frame D that depends on missed p-Frame C then will not be able to reflect changes in p-Frame C. At best, unless Frames B and C were very similar this will result in some jerkiness or noise in the video signal, while at worst this may result in a video frame for p-Frame D being unable to be properly decoded and provided for display. Similarly, subsequent frames such as p-Frame E will be unable to recover this lost information, and may continue the problems with the displayed video based, at least in part, upon the lost change data for p-Frame C. If a server sending the video stream in this example does not receive an acknowledgement for p-Frame C then the server can determine that a new i-Frame F needs to be transmitted in order to recover from the loss of data. In other examples, the client receiving the sequence may realize that a frame was dropped, and may request a new i-Frame. While the transmission of a new i-

Frame enables the video to recover as if no data had been lost, the generation of a new i-Frame takes some time, and the transmission of the relatively large amount of data for a new i-Frame can also take some time (and additional bandwidth), such that there will be some delay between the dropped frame and a new i-Frame F being sent. For applications that are time-critical or at least latency sensitive, such delay or latency can have significant negative impacts, as discussed previously.

[0027] Certain conventional approaches rely upon client-side determinations that a frame in a sequence or cadence was not received within an expected time window. For example, if a system is running at a certain frame rate, such as 60 FPS, then a client device will know approximately when to expect the next frame in a video sequence to be received from across a network. With a bit of tuning, the client device can determine how long to wait for a given frame to arrive. If a frame does not arrive within that period of time, it can be assumed for many situations that the data for that frame is lost. Such an approach can be very expensive, however, as there is generally some lag in getting that information back to the server, having the server process that data, and then send new, updated, or additional data as a result. Further, if the stream is invalidated by the client, then any subsequent progressive frames that have been, or will be, received after that point will be invalid, such that the client will have to wait until another i-Frame is received to effectively restart the video stream from the current point.

[0028] Some conventional approaches utilize some type of forward error correction (FEC) to attempt to minimize the impact of packet loss during data transmission. FEC effectively causes the source of the transmission, such as a media server, to send redundant data, whereby the receiver of the transmission, such as a client device, can utilize only the portion of the data is received and is error-free. Such an approach can enable the client to reconstruct at least some of the data that was missing using this extra, or redundant, data. For limited data loss, a client may be able to fully recover without receiving the lost packet(s). Such an approach is not optimal, however, as it requires additional bandwidth for this redundant data, and is still susceptible to packet loss if the loss is not small enough to allow for recovery.

[0029] Accordingly, approaches in accordance with various embodiments can compress or encode video sequences, or other such data, in such a way as to avoid at least some of these and other such issues with respect to data loss or corruption. Such an approach can utilize modified compression approaches, as may be based at least in part upon encoding formats such as H.264, HEVC, VP9, FLV, or WEBM, among others. FIG. 3A illustrates an initial set of frames 300 of an example video sequence 302 that can be transmitted in accordance with various embodiments. In this example, an i-Frame A is transmitted as a first frame in this sequence, at least because there is no prior frame in this sequence to refer to, or depend from. This i-Frame can include a full set of image data, including values for each pixel in the frame. A next frame in the sequence 302, here p-Frame B, can include data for changes with respect to prior frame A, similar to the sequence described with respect to FIG. 2A. In this example, however, a server transmitting this sequence (or other such source) may not yet have received an acknowledgement that p-Frame B was properly and completely received and decoded. Accordingly, the server may determine not to generate data for the next frame in the sequence, here p-Frame C, to include changes with respect to immediately prior frame B, but instead to the last acknowledged “good” frame, here i-Frame A. In this way, p-Frame C may involve more data to be transmitted than if p-Frame C only reflected changes with respect to p-Frame B, but the data for p-Frame C will allow p-Frame C to be fully and accurately decoded and displayed even if intermediate p-Frame B was dropped or corrupted. Similarly, subsequent p-Frame D in this example can also involve data for changes with respect to last acknowledged frame A, so that frame D could be properly displayed, without noticeable added latency, even if frames B and/or C were dropped or corrupted.

[0030] In this example, an acknowledgement (or “ack”) is received for p-Frame B after at least the generation of p-Frame D. As illustrated, the next frame (p-Frame E) can then have the data determined to transmit that represent changes with respect to the last acknowledged good frame, which is now p-Frame B as illustrated in FIG. 3B. If no acknowledgement is received for frames C, D, or E before the data is determined to transmit for p-Frame F, then p-Frame F can also have data that represents changes with respect to last acknowledged frame B. If, however, an acknowledgement is received for frame C, D, or E before the data for frame F is determined, then that data can then instead represent changes with respect to the most recent frame (in the

sequence 302) among those frames for which an acknowledgement was received. In many embodiments, frame data is sent at a regular interval which may approximate the frame rate at which those frames are to be displayed, such as sixty frames per second (fps). Accordingly, acknowledgement of frames should be received with a similar frequency, such that unless a frame is dropped most frames may end up depending upon a most recent frame. If an acknowledgement is not received for a duration corresponding to several frames, then that may be indicative of a problem that may need to be addressed using another approach. In such an instance, sending a new i-Frame may not solve the problem as that frame may not be properly or timely received either. In some embodiments, an i-Frame may be sent after a determined number of frames have been transmitted without an acknowledgement, such as five or more frames.

[0031] As illustrated in FIG. 3C, if a frame such as p-Frame C is dropped (or corrupted, etc.), then subsequent p-Frame D can still represent changes with respect to last acknowledged frame A. In some instances, the frame data may have been properly received but the corresponding acknowledgement may not have been properly received. Once another frame, here p-Frame B, has an acknowledgement received, then subsequent frames can depend upon that frame, as illustrated for p-Frames E and F. This approach effectively creates a sliding window where any given frame will depend from a most recent frame in the sequence for which an acknowledgement has been received (or other indication provided or determined). As illustrated in FIG. 3D, subsequent frames can depend upon p-Frame D once an acknowledgement is received, and so on. In this way, even if one or more frames are dropped the subsequent frames can recover and decode or display proper images by utilizing data that is representative of changes (*e.g.*, differentials) in the video content since a last acknowledged good frame. A dropped frame may then result in only a temporary delay (on the order of a frame) in updating of the video, which may result in a jump in pixel values that may be barely noticeable, and may be sufficiently short as to have minimal, if any, impact on the application. For example, a gamer may not be unfairly disadvantaged by not having the video update for a period of time on the order of about 30 milliseconds, but might be disadvantaged if a number of frames were dropped or corrupted and a new i-Frame needed to be sent, as in prior methods, which might result in a

delay on the order of a second or more. Forward error correction can still be used with such a process.

[0032] In at least one embodiment, frames received by a client device can be in a confirmed state or an unconfirmed state, which may correspond to whether a frame has been properly received, decoded, verified, and/or acknowledged. When a frame is successfully decoded and acknowledged, for example, that frame can be considered to be a confirmed frame. In at least one embodiment, a client device might keep only the most recent confirmed frame in a local buffer. This confirmed frame will correspond to the last good frame of a sequence, and can be used to decode subsequent frames until a new frame in that sequence is confirmed.

[0033] In at least one embodiment, a client device might cache at least the two most recent confirmed frames in the buffer. This can help account for the asynchronous nature of communications, in case a frame is acknowledged by the client, but that acknowledgement is not received and processed by the server before a next frame is generated and transmitted, which would then represent a differential from the prior confirmed frame. In this way, a frame that is received with differential data with respect to either prior confirmed frame can be properly decoded and displayed, and verified as a new confirmed frame.

[0034] In some embodiments, a client device might buffer multiple confirmed frames, such that any subsequent progressive frame can utilize, as a base for the differential, any of those buffered confirmed reference frames. In at least one embodiment, a transmitting server can maintain a list of confirmed reference frames buffered by a client device, and can update this list each time a new confirmed frame is added to, or removed from, this buffer. In at least one embodiment, a client device might send a request to the server that a confirmed frame be removed from the buffer. The client device can then wait for an acknowledgement back from the server before removing that confirmed frame from the buffer. If an acknowledgement from the server is not received after a period of time, the client device can send another request to remove that frame from the buffer. Such an approach can prevent overrun of the buffer. Such an approach can also help to avoid the situation where the server lists a given frame being available as a potential reference frame but that frame is no longer available on the client device. Further,

this helps to avoid the situation where a frame is already in the process of being transmitted as the frame is removed from the buffer, resulting in that confirmed frame no longer being available as a reference frame. In some embodiments, the client device might request that the server select a frame to be removed from the buffer, in which case the server can respond with an instruction to remove one or more selected frames from the buffer, which the server will no longer use as reference frames. This can enable the server to have management responsibility for the buffer on the client device in at least some embodiments.

[0035] In at least one embodiment, a transmitting server or source can attempt to determine which confirmed frame to use as a reference frame for a subsequent progressive frame. There can be various rules or policies set in place to make such determinations, as may be user-configurable or may be made based at least in part upon network conditions. In at least one embodiment, a reference frame can be selected that would result in the smallest differential, such as may have the smallest number of changes or may result in the least amount of data to be transmitted for a subsequent frame. In some embodiments, where there are no significant network issues or concerns, a reference frame might instead be selected that includes the most changes in order to minimize the impact of any corrupted or incorrect data in a prior frame. In some embodiments, a transmitting server might have a preference to utilize more recent confirmed frames in order to avoid a situation where an older frame might be removed from the buffer on the client device. Other rules or approaches for selection can be used as well within the scope of the various embodiments.

[0036] In one such implementation, a client device could inform the server when a confirmed frame is flushed from the buffer, such that the server knows that frame can no longer be used as a base, reference, or key frame for subsequent frames. In some embodiments, the server could instruct the client device as to which confirmed frames to remove from the buffer, such as may be based upon the differences exceeding a maximum differential threshold or having a maximum number of frames in the buffer with a smaller differential with respect to a current frame, among other such options. A client device could also ask the server which frame to discard from the buffer if the client needs or wants to clear space in the buffer. In some embodiments, at least

some of this information can be conveyed through the acknowledgement messages sent from the client device or other such recipient.

[0037] Subsequent frames that have not been confirmed can also be stored in the buffer, as these frames may become new confirmed frames in the sequence after acknowledgement is sent or another confirmation requirement is satisfied. Each time a new frame is confirmed, one of the older confirmed frames in the sequence may be discarded from the buffer, at least when a maximum number or size of frames in the buffer is met, approached, or exceeded. In some embodiments, a maximum number of unconfirmed frames may be kept in the buffer, as a lengthy period without an acknowledgement or confirmation may be indicative of network or other problems, such that a different remedial action should be taken or problem investigated. In many streams, a new i-Frame or key frame may be generated for changes such as scene or programming changes, or new levels or cutscenes in a game, etc. (where most if not all of the pixel data will change), and any time a new i-Frame is received by a client the client may flush all prior frames from the buffer as they will no longer be able to serve as key frames for subsequent frames in that video sequence. In at least some embodiments, new i-Frames are only sent when necessary or requested, such that there may be very few (as few as one) i-Frames sent for a given stream, which can be beneficial since i-Frames can be expensive due to the amount of data to be generated, transmitted, and stored. If the problem is a network bandwidth problem, sending unnecessary i-Frames can actually exacerbate the problem, particularly for higher resolutions such as 4K or 8K.

[0038] As mentioned, benefits of such approaches can be used for applications and use cases beyond video and media applications. Advantages can be obtained for various use cases where differentiable or reference data is used, where a reference point can be advantageously updated over time. This can include, for example, differentiable temperature or sensor data transmission for mechanical or electrical systems. Such an approach can also be used advantageously for use cases where latency and buffering are an issue, as may relate to videoconferencing, audio streaming, automation, and the like. Even for a use case such as online gaming where video may benefit from such an approach, there may be other data or information that can also benefit. For example, input data (*e.g.*, user input data from a mouse, control pad, VR control stick, audio

input device, motion sensor, gesture control, or touch screen, among other such options) that is sent from a client device to a server when streaming games is often sent as a set of relative positional movements. This data can have a similar need for low latency, and it can be desirable to avoid this input data getting corrupted as a result of missing data. By coalescing events (or buffering data for each event) since the last acknowledged event, benefits similar to those for the video stream can be obtained, such as by handling missed packets without the overhead of requesting a resend of the data. Such an approach can apply to the transmission of an individual "state" or "timeslice" of a changing data set, for example, where a subsequent state or timeslice of the data set may be encoded, or at least described or generated, as a set of differentials versus a specific previous state or timeslice of the same data set.

[0039] FIG. 4 illustrates an example video encoding pipeline 400 that can be used to transmit video data to at least one client device, or other such recipient, for presentation. It should be understood that such video data can include, or be associated with, audio data as well in at least some embodiments. In this example, pixel data for a current frame to be rendered can be received from a video source 402, such as a camera, media server, or online game host. In this example, the video data is received to a video encoder 404 that is configured to encode the data using a particular encoding format. For an initial frame, the encoder 404 can encode this frame as an i-Frame, or key frame, that can be passed to a stream server 408, or other such transmitter, to be transmitted across at least one network 410 to a client device 412 or other intended recipient. A receiver 414 of the client device can receive this video data, pass the data to a video decoder 416 to decode the video frame, and can then cause the decoded video frame to be presented via a display 420 or other such mechanism, such as a projector, monitor, or wearable display. In this example, the video decoder 416 can cause this frame to be stored to a reference buffer 418 as a confirmed frame that can be used as a reference for subsequent frames in this video stream. The video decoder 416 can also cause an acknowledgement to be sent back to the stream server 408, which can communicate the acknowledgement to the video encoder 404. The video encoder can have stored a copy of that i-frame in a reference buffer 406, and can designate that frame as a confirmed frame that can now be used as a basis for subsequent video frames on this stream.

[0040] In this example, a second frame in this sequence can use the initial i-Frame as a reference frame because there are no other frames at this time to serve as a reference frame for the second frame, whether an acknowledgement has been received for the first frame yet or not. For a third or subsequent frame in this sequence, the video encoder 404 can check the confirmed reference frames in the reference buffer 406 to determine which to use as a reference frame for determining a differential for that current frame. As mentioned, this selection can be determined using any of a number of different rules, policies, or approaches, such as to select the reference frame that would result in the smallest number of changes or would result in the least amount of data to be transmitted for the current frame. In at least one embodiment, the video encoder can select as a reference frame any frame in the local reference buffer 406 that has been confirmed by the client device 412 and indicated to be cached or buffered in the client reference buffer 418. The video encoder 404, once a reference frame has been selected, can generate the differential and encode data for the current video frame that can be transmitted to the client device 412. Once properly decoded and verified in this example, the client device 412 can cause the current frame to be stored to the client reference buffer 418 and an acknowledgement sent to the stream server 408 that the new frame may now serve as a reference frame for subsequent frames on that stream.

[0041] If a buffer removal criterion is met, such as a maximum number of buffered frames met or exceeded, then the client device 412 can submit a request to the stream server 408 or video encoder 404 to remove at least one older or prior confirmed frame in the sequence from the client reference buffer 418. An acknowledgement can be received from the video encoder 404 and/or stream server acknowledging that one or more specific frames can be removed from the client reference buffer 418 and will no longer be used as a reference frame. As mentioned, this can include one or more frames requested for removal by the client device or selected for removal by, for example, the video encoder 404. In at least one embodiment, the video encoder 404 (or other responsible or monitoring component) can instruct that a confirmed frame is to be removed from the buffer without receiving a request from the client device. In at least one embodiment, a frame other than the oldest frame may be removed from the reference buffer, such as to remove a frame that is most similar to another confirmed frame in the buffer, in order

to maximize a diversity of data in the buffer to allow for better compression optimization for subsequent frames. In various embodiments, the video encoder 404 may encode multiple streams to be transmitted to multiple recipients, where the progressive frame sent on each stream may depend at least in part to the confirmed frames available on those devices. In some embodiments, a maximum number of different reference frames might be allowed to be used by the video encoder, in order to prevent significant additional processing requirements on the video encoder, as well as to limit a size of the reference buffer 406.

[0042] FIG. 5A illustrates an example process 500 for encoding a data stream for transmission that can be performed in accordance with various embodiments. In this example, an initial video frame in a sequence is encoded 502 as a key frame. This may be a first frame in an overall sequence, or a first frame after a scene change, among other such options. This initial key frame can be buffered 504 as a confirmed frame, or a frame that is otherwise confirmed to be able to be used as a reference frame for subsequent frames. In at least one embodiment, every key frame or i-Frame may be designated as a confirmed frame that can be used as a reference frame, at least because there may currently be no other frame to be used as a reference frame, and it may be undesirable to send another key frame unless a confirmation or acknowledgement is first received. This key frame can also be transmitted 506 to a client device, or other such recipient, for decoding and display.

[0043] For subsequent frames in this sequence, a confirmed frame from the buffered frames can be determined 508 or selected for use as a reference frame. This can include, for example, determining a confirmed frame that would result in the smallest differential or least number of changes with respect to a frame to be encoded. In at least one embodiment, this may include determining two or more differentials in parallel, and then selecting one of the confirmed frames based at least in part upon one or more aspects of those differentials. The next frame can then be encoded 510 using the differential data with respect to the selected reference frame. Data for this frame can be stored to the buffer for potential use as a future reference frame for the sequence. The encoded data for this frame can also be transmitted 514 to the client device for decoding and display as part of the current video sequence. During this process, a determination can be made 516 as to whether there has been a change to the buffer on the client device. This can include,

for example, receiving an acknowledgement from the client device of a new confirmed frame in the buffer, or notification of removal of one or more frames from the client reference buffer. In embodiments where a video encoder may instruct the client device to remove frames from the client reference buffer, this may include notification of such instruction being sent. If there is no change to the buffer, then the process can continue. If there is a change to the client reference buffer, then the frame data stored in the local buffer can be updated 518 to reflect the change(s), such as to reflect the confirmed frames that are currently buffered on the client device and available to serve as a reference frame for one or more subsequent frames in the sequence. The process can then continue based at least in part upon this updated buffer data.

[0044] FIG. 5B illustrates a process 550 for decoding and displaying video data that can be performed in accordance with various embodiments. In this example, encoded data is received 552 for a next video frame in a sequence, such as a frame subsequent to an initial key frame that was received, decoded, and displayed. This frame data can be decoded 554 using data for an identified reference frame (unless the frame is a key frame without a reference frame) that is stored in a client reference buffer. The identified reference frame can refer to a prior frame in this sequence that was properly received and decoded, and thus can serve as a reference frame for purposes of decoding a full video frame using differential video data between these frames. This successfully decoded video frame can be provided 556 for presentation, such as for display through an appropriate display element. Data for this successfully decoded frame can also be stored 558 in the client reference buffer for potential use as a reference frame for one or more subsequent frames in this sequence. An acknowledgement can also be sent 560 indicating that the frame was successfully decoded and is stored in the client reference buffer and available as a potential reference frame. During this process, a determination can be made 562 to remove one or more frames from the client reference buffer. As mentioned, this decision can be made by the client device, video encoder, or a remote server, among other such options. If the client device determines that at least one frame is to be removed, a request can be sent 564 to the video encoder, or a component or server managing the encoding process, to remove one or more from the client reference buffer. The request can specify the frame(s), or allow the video encoder (or other component) to select the frame(s), to be removed. Once an acknowledgement is received,

the specified frame(s) can be removed 566 from the client reference buffer and will be unavailable for use as a potential reference frame for future frames on the sequence. As mentioned, in other embodiments the video encoder may instruct that one or more frames be removed from the client buffer, rather than responding to a request from the client device. The process can then continue using the current frames in the client reference buffer.

[0045] As discussed, various approaches presented herein are lightweight enough to execute on a client device, such as a personal computer or gaming console, in real time. Such processing can be performed on content that is generated on that client device or received from an external source, such as streaming content received over at least one network. The source can be any appropriate source, such as a game host, streaming media provider, third party content provider, or other client device, among other such options. In some instances, the processing and/or rendering of this content may be performed by one of these other devices, systems, or entities, then provided to the client device (or another such recipient) for presentation or another such use.

[0046] As an example, FIG. 6 illustrates an example network configuration 600 that can be used to provide, generate, modify, encode, and/or transmit content. In at least one embodiment, a client device 602 can generate or receive content for a session using components of a content application 604 on client device 602 and data stored locally on that client device. In at least one embodiment, a content application 624 (*e.g.*, an image generation or editing application) executing on content server 620 (*e.g.*, a cloud server or edge server) may initiate a session associated with at least client device 602, as may utilize a session manager and user data stored in a user database 634, and can cause content 632 to be determined by a content manager 626. A scene generation module 628, as may relate to an animation or gaming application, may generate or obtain content determined to be provided, where at least a portion of that content may need to be rendered using a rendering engine, if needed for this type of content or platform, and transmitted to client device 602 using an appropriate transmission manager 622 to send by download, streaming, or another such transmission channel. An encoder 630 can be used to encode and/or compress this data before transmitting to the client device 602. In at least one embodiment, this content 632 can include video or image data for a scene. In at least one embodiment, client device 602 receiving this content can provide this content to a corresponding

content application 604, which may also or alternatively include a scene generation module or rendering engine 612 (if necessary) for rendering at least some of this content for presentation. A decoder 614 can also be used to decode data received over the network(s) 640 for presentation via client device 602, such as image or video content through a display 606 and audio, such as sounds and music, through at least one audio playback device 608, such as speakers or headphones. In at least one embodiment, at least some of this content may already be stored on, rendered on, or accessible to client device 602 such that transmission over network 640 is not required for at least that portion of content, such as where that content may have been previously downloaded or stored locally on a hard drive or optical disk. In at least one embodiment, a transmission mechanism such as data streaming can be used to transfer this content from server 620, or content database 634, to client device 602. In at least one embodiment, at least a portion of this content can be obtained or streamed from another source, such as a third party content service 660 that may also include a content application 662 for generating or providing content. In at least one embodiment, portions of this functionality can be performed using multiple computing devices, or multiple processors within one or more computing devices, such as may include a combination of CPUs and GPUs.

[0047] In at least one embodiment, content application 624 includes a content manager 626 that can determine or analyze content before this content is transmitted to client device 602. In at least one embodiment, content manager 626 can also include, or work with, other components that are able to generate, modify, encode, or enhance content to be provided. In at least one embodiment, this can include determining which frames are available to serve as reference frames, or select reference frames for specific frames to be encoded. In at least one embodiment, an image, video, or scene generation component 628 can be used to generate image, video, or other media content. In at least one embodiment, content manager 626 can cause selected and encoded content to be transmitted to client device 602. In at least one embodiment, a content application 604 on client device 602 may also include components such as a rendering engine or image or video decoder 614, such that any or all of this functionality can additionally, or alternatively, be performed on client device 602. In at least one embodiment, a content application 662 on a third party content service system 660 can also include such functionality.

In at least one embodiment, locations where at least some of this functionality is performed may be configurable, or may depend upon factors such as a type of client device 602 or availability of a network connection with appropriate bandwidth, among other such factors. In at least one embodiment, a system for content generation can include any appropriate combination of hardware and software in one or more locations. In at least one embodiment, generated image or video content of one or more resolutions can also be provided, or made available, to other client devices 650, such as for download or streaming from a media source storing a copy of that image or video content. In at least one embodiment, this may include transmitting images of game content for a multiplayer game, where different client devices may display that content at different resolutions, including one or more super-resolutions.

[0048] In this example, these client devices can include any appropriate computing devices, as may include a desktop computer, notebook computer, set-top box, streaming device, gaming console, smartphone, tablet computer, VR headset, AR goggles, wearable computer, or a smart television. Each client device can submit a request across at least one wired or wireless network, as may include the Internet, an Ethernet, a local area network (LAN), or a cellular network, among other such options. In this example, these requests can be submitted to an address associated with a cloud provider, who may operate or control one or more electronic resources in a cloud provider environment, such as may include a data center or server farm. In at least one embodiment, the request may be received or processed by at least one edge server, that sits on a network edge and is outside at least one security layer associated with the cloud provider environment. In this way, latency can be reduced by enabling the client devices to interact with servers that are in closer proximity, while also improving security of resources in the cloud provider environment.

[0049] In at least one embodiment, such a system can be used for performing graphical rendering operations. In other embodiments, such a system can be used for other purposes, such as for providing image or video content to test or validate autonomous machine applications, or for performing deep learning operations. In at least one embodiment, such a system can be implemented using an edge device, or may incorporate one or more Virtual Machines (VMs). In

at least one embodiment, such a system can be implemented at least partially in a data center or at least partially using cloud computing resources.

INFERENCE AND TRAINING LOGIC

[0050] FIG. 7A illustrates inference and/or training logic 715 used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGS. 7A and/or 7B.

[0051] In at least one embodiment, inference and/or training logic 715 may include, without limitation, code and/or data storage 701 to store forward and/or output weight and/or input/output data, and/or other parameters to configure neurons or layers of a neural network trained and/or used for inferencing in aspects of one or more embodiments. In at least one embodiment, training logic 715 may include, or be coupled to code and/or data storage 701 to store graph code or other software to control timing and/or order, in which weight and/or other parameter information is to be loaded to configure, logic, including integer and/or floating point units (collectively, arithmetic logic units (ALUs)). In at least one embodiment, code, such as graph code, loads weight or other parameter information into processor ALUs based on an architecture of a neural network to which the code corresponds. In at least one embodiment, code and/or data storage 701 stores weight parameters and/or input/output data of each layer of a neural network trained or used in conjunction with one or more embodiments during forward propagation of input/output data and/or weight parameters during training and/or inferencing using aspects of one or more embodiments. In at least one embodiment, any portion of code and/or data storage 701 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0052] In at least one embodiment, any portion of code and/or data storage 701 may be internal or external to one or more processors or other hardware logic devices or circuits. In at least one embodiment, code and/or code and/or data storage 701 may be cache memory, dynamic randomly addressable memory ("DRAM"), static randomly addressable memory ("SRAM"), non-volatile memory (e.g., Flash memory), or other storage. In at least one embodiment, choice of whether code and/or code and/or data storage 701 is internal or external to a processor, for

example, or comprised of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors.

[0053] In at least one embodiment, inference and/or training logic 715 may include, without limitation, a code and/or data storage 705 to store backward and/or output weight and/or input/output data corresponding to neurons or layers of a neural network trained and/or used for inferencing in aspects of one or more embodiments. In at least one embodiment, code and/or data storage 705 stores weight parameters and/or input/output data of each layer of a neural network trained or used in conjunction with one or more embodiments during backward propagation of input/output data and/or weight parameters during training and/or inferencing using aspects of one or more embodiments. In at least one embodiment, training logic 715 may include, or be coupled to code and/or data storage 705 to store graph code or other software to control timing and/or order, in which weight and/or other parameter information is to be loaded to configure, logic, including integer and/or floating point units (collectively, arithmetic logic units (ALUs)). In at least one embodiment, code, such as graph code, loads weight or other parameter information into processor ALUs based on an architecture of a neural network to which the code corresponds. In at least one embodiment, any portion of code and/or data storage 705 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory. In at least one embodiment, any portion of code and/or data storage 705 may be internal or external to on one or more processors or other hardware logic devices or circuits. In at least one embodiment, code and/or data storage 705 may be cache memory, DRAM, SRAM, non-volatile memory (e.g., Flash memory), or other storage. In at least one embodiment, choice of whether code and/or data storage 705 is internal or external to a processor, for example, or comprised of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors.

[0054] In at least one embodiment, code and/or data storage 701 and code and/or data storage 705 may be separate storage structures. In at least one embodiment, code and/or data storage 701 and code and/or data storage 705 may be same storage structure. In at least one embodiment, code and/or data storage 701 and code and/or data storage 705 may be partially same storage structure and partially separate storage structures. In at least one embodiment, any portion of code and/or data storage 701 and code and/or data storage 705 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0055] In at least one embodiment, inference and/or training logic 715 may include, without limitation, one or more arithmetic logic unit(s) ("ALU(s)") 710, including integer and/or floating point units, to perform logical and/or mathematical operations based, at least in part on, or indicated by, training and/or inference code (e.g., graph code), a result of which may produce activations (e.g., output values from layers or neurons within a neural network) stored in an activation storage 720 that are functions of input/output and/or weight parameter data stored in code and/or data storage 701 and/or code and/or data storage 705. In at least one embodiment, activations stored in activation storage 720 are generated according to linear algebraic and or matrix-based mathematics performed by ALU(s) 710 in response to performing instructions or other code, wherein weight values stored in code and/or data storage 705 and/or code and/or data storage 701 are used as operands along with other values, such as bias values, gradient information, momentum values, or other parameters or hyperparameters, any or all of which may be stored in code and/or data storage 705 or code and/or data storage 701 or another storage on or off-chip.

[0056] In at least one embodiment, ALU(s) 710 are included within one or more processors or other hardware logic devices or circuits, whereas in another embodiment, ALU(s) 710 may be external to a processor or other hardware logic device or circuit that uses them (e.g., a co-processor). In at least one embodiment, ALUs 710 may be included within a processor's execution units or otherwise within a bank of ALUs accessible by a processor's execution units either within same processor or distributed between different processors of different types (e.g., central processing units, graphics processing units, fixed function units, etc.). In at least one

embodiment, code and/or data storage 701, code and/or data storage 705, and activation storage 720 may be on same processor or other hardware logic device or circuit, whereas in another embodiment, they may be in different processors or other hardware logic devices or circuits, or some combination of same and different processors or other hardware logic devices or circuits. In at least one embodiment, any portion of activation storage 720 may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory. Furthermore, inferencing and/or training code may be stored with other code accessible to a processor or other hardware logic or circuit and fetched and/or processed using a processor's fetch, decode, scheduling, execution, retirement and/or other logical circuits.

[0057] In at least one embodiment, activation storage 720 may be cache memory, DRAM, SRAM, non-volatile memory (e.g., Flash memory), or other storage. In at least one embodiment, activation storage 720 may be completely or partially within or external to one or more processors or other logical circuits. In at least one embodiment, choice of whether activation storage 720 is internal or external to a processor, for example, or comprised of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors. In at least one embodiment, inference and/or training logic 715 illustrated in FIG. 7a may be used in conjunction with an application-specific integrated circuit ("ASIC"), such as Tensorflow® Processing Unit from Google, an inference processing unit (IPU) from Graphcore™, or a Nervana® (e.g., "Lake Crest") processor from Intel Corp. In at least one embodiment, inference and/or training logic 715 illustrated in FIG. 7a may be used in conjunction with central processing unit ("CPU") hardware, graphics processing unit ("GPU") hardware or other hardware, such as field programmable gate arrays ("FPGAs").

[0058] FIG. 7b illustrates inference and/or training logic 715, according to at least one or more embodiments. In at least one embodiment, inference and/or training logic 715 may include, without limitation, hardware logic in which computational resources are dedicated or otherwise exclusively used in conjunction with weight values or other information corresponding to one or more layers of neurons within a neural network. In at least one embodiment, inference and/or

training logic 715 illustrated in FIG. 7b may be used in conjunction with an application-specific integrated circuit (ASIC), such as Tensorflow® Processing Unit from Google, an inference processing unit (IPU) from Graphcore™, or a Nervana® (e.g., “Lake Crest”) processor from Intel Corp. In at least one embodiment, inference and/or training logic 715 illustrated in FIG. 7b may be used in conjunction with central processing unit (CPU) hardware, graphics processing unit (GPU) hardware or other hardware, such as field programmable gate arrays (FPGAs). In at least one embodiment, inference and/or training logic 715 includes, without limitation, code and/or data storage 701 and code and/or data storage 705, which may be used to store code (e.g., graph code), weight values and/or other information, including bias values, gradient information, momentum values, and/or other parameter or hyperparameter information. In at least one embodiment illustrated in FIG. 7b, each of code and/or data storage 701 and code and/or data storage 705 is associated with a dedicated computational resource, such as computational hardware 702 and computational hardware 706, respectively. In at least one embodiment, each of computational hardware 702 and computational hardware 706 comprises one or more ALUs that perform mathematical functions, such as linear algebraic functions, only on information stored in code and/or data storage 701 and code and/or data storage 705, respectively, result of which is stored in activation storage 720.

[0059] In at least one embodiment, each of code and/or data storage 701 and 705 and corresponding computational hardware 702 and 706, respectively, correspond to different layers of a neural network, such that resulting activation from one “storage/computational pair 701/702” of code and/or data storage 701 and computational hardware 702 is provided as an input to “storage/computational pair 705/706” of code and/or data storage 705 and computational hardware 706, in order to mirror conceptual organization of a neural network. In at least one embodiment, each of storage/computational pairs 701/702 and 705/706 may correspond to more than one neural network layer. In at least one embodiment, additional storage/computation pairs (not shown) subsequent to or in parallel with storage computation pairs 701/702 and 705/706 may be included in inference and/or training logic 715.

DATA CENTER

[0060] FIG. 8 illustrates an example data center 800, in which at least one embodiment may be used. In at least one embodiment, data center 800 includes a data center infrastructure layer 810, a framework layer 820, a software layer 830, and an application layer 840.

[0061] In at least one embodiment, as shown in FIG. 8, data center infrastructure layer 810 may include a resource orchestrator 812, grouped computing resources 814, and node computing resources (“node C.R.s”) 816(1)-816(N), where “N” represents any whole, positive integer. In at least one embodiment, node C.R.s 816(1)-816(N) may include, but are not limited to, any number of central processing units (“CPUs”) or other processors (including accelerators, field programmable gate arrays (FPGAs), graphics processors, etc.), memory devices (e.g., dynamic read-only memory), storage devices (e.g., solid state or disk drives), network input/output (“NW I/O”) devices, network switches, virtual machines (“VMs”), power modules, and cooling modules, etc. In at least one embodiment, one or more node C.R.s from among node C.R.s 816(1)-816(N) may be a server having one or more of above-mentioned computing resources.

[0062] In at least one embodiment, grouped computing resources 814 may include separate groupings of node C.R.s housed within one or more racks (not shown), or many racks housed in data centers at various geographical locations (also not shown). Separate groupings of node C.R.s within grouped computing resources 814 may include grouped compute, network, memory or storage resources that may be configured or allocated to support one or more workloads. In at least one embodiment, several node C.R.s including CPUs or processors may grouped within one or more racks to provide compute resources to support one or more workloads. In at least one embodiment, one or more racks may also include any number of power modules, cooling modules, and network switches, in any combination.

[0063] In at least one embodiment, resource orchestrator 812 may configure or otherwise control one or more node C.R.s 816(1)-816(N) and/or grouped computing resources 814. In at least one embodiment, resource orchestrator 812 may include a software design infrastructure (“SDI”) management entity for data center 800. In at least one embodiment, resource orchestrator may include hardware, software or some combination thereof.

[0064] In at least one embodiment, as shown in FIG. 8, framework layer 820 includes a job scheduler 822, a configuration manager 824, a resource manager 826 and a distributed file system 828. In at least one embodiment, framework layer 820 may include a framework to support software 832 of software layer 830 and/or one or more application(s) 842 of application layer 840. In at least one embodiment, software 832 or application(s) 842 may respectively include web-based service software or applications, such as those provided by Amazon Web Services, Google Cloud and Microsoft Azure. In at least one embodiment, framework layer 820 may be, but is not limited to, a type of free and open-source software web application framework such as Apache Spark™ (hereinafter “Spark”) that may utilize distributed file system 828 for large-scale data processing (e.g., “big data”). In at least one embodiment, job scheduler 822 may include a Spark driver to facilitate scheduling of workloads supported by various layers of data center 800. In at least one embodiment, configuration manager 824 may be capable of configuring different layers such as software layer 830 and framework layer 820 including Spark and distributed file system 828 for supporting large-scale data processing. In at least one embodiment, resource manager 826 may be capable of managing clustered or grouped computing resources mapped to or allocated for support of distributed file system 828 and job scheduler 822. In at least one embodiment, clustered or grouped computing resources may include grouped computing resource 814 at data center infrastructure layer 810. In at least one embodiment, resource manager 826 may coordinate with resource orchestrator 812 to manage these mapped or allocated computing resources.

[0065] In at least one embodiment, software 832 included in software layer 830 may include software used by at least portions of node C.R.s 816(1)-816(N), grouped computing resources 814, and/or distributed file system 828 of framework layer 820. The one or more types of software may include, but are not limited to, Internet web page search software, e-mail virus scan software, database software, and streaming video content software.

[0066] In at least one embodiment, application(s) 842 included in application layer 840 may include one or more types of applications used by at least portions of node C.R.s 816(1)-816(N), grouped computing resources 814, and/or distributed file system 828 of framework layer 820. One or more types of applications may include, but are not limited to, any number of a genomics

application, a cognitive compute, and a machine learning application, including training or inferencing software, machine learning framework software (e.g., PyTorch, TensorFlow, Caffe, etc.) or other machine learning applications used in conjunction with one or more embodiments.

[0067] In at least one embodiment, any of configuration manager 824, resource manager 826, and resource orchestrator 812 may implement any number and type of self-modifying actions based on any amount and type of data acquired in any technically feasible fashion. In at least one embodiment, self-modifying actions may relieve a data center operator of data center 800 from making possibly bad configuration decisions and possibly avoiding underutilized and/or poor performing portions of a data center.

[0068] In at least one embodiment, data center 800 may include tools, services, software or other resources to train one or more machine learning models or predict or infer information using one or more machine learning models according to one or more embodiments described herein. For example, in at least one embodiment, a machine learning model may be trained by calculating weight parameters according to a neural network architecture using software and computing resources described above with respect to data center 800. In at least one embodiment, trained machine learning models corresponding to one or more neural networks may be used to infer or predict information using resources described above with respect to data center 800 by using weight parameters calculated through one or more training techniques described herein.

[0069] In at least one embodiment, data center may use CPUs, application-specific integrated circuits (ASICs), GPUs, FPGAs, or other hardware to perform training and/or inferencing using above-described resources. Moreover, one or more software and/or hardware resources described above may be configured as a service to allow users to train or performing inferencing of information, such as image recognition, speech recognition, or other artificial intelligence services.

[0070] Inference and/or training logic 715 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGs. 7A and/or 7B. In at least one

embodiment, inference and/or training logic 715 may be used in system FIG. 8 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0071] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

COMPUTER SYSTEMS

[0072] FIG. 9 is a block diagram illustrating an exemplary computer system, which may be a system with interconnected devices and components, a system-on-a-chip (SOC) or some combination thereof 900 formed with a processor that may include execution units to execute an instruction, according to at least one embodiment. In at least one embodiment, computer system 900 may include, without limitation, a component, such as a processor 902 to employ execution units including logic to perform algorithms for process data, in accordance with present disclosure, such as in embodiment described herein. In at least one embodiment, computer system 900 may include processors, such as PENTIUM® Processor family, Xeon™, Itanium®, XScale™ and/or StrongARM™, Intel® Core™, or Intel® Nervana™ microprocessors available from Intel Corporation of Santa Clara, California, although other systems (including PCs having other microprocessors, engineering workstations, set-top boxes and like) may also be used. In at least one embodiment, computer system 900 may execute a version of WINDOWS' operating system available from Microsoft Corporation of Redmond, Wash., although other operating systems (UNIX and Linux for example), embedded software, and/or graphical user interfaces, may also be used.

[0073] Embodiments may be used in other devices such as handheld devices and embedded applications. Some examples of handheld devices include cellular phones, Internet Protocol devices, digital cameras, personal digital assistants ("PDAs"), and handheld PCs. In at least one embodiment, embedded applications may include a microcontroller, a digital signal processor ("DSP"), system on a chip, network computers ("NetPCs"), set-top boxes, network hubs, wide

area network (“WAN”) switches, or any other system that may perform one or more instructions in accordance with at least one embodiment.

[0074] In at least one embodiment, computer system 900 may include, without limitation, processor 902 that may include, without limitation, one or more execution units 908 to perform machine learning model training and/or inferencing according to techniques described herein. In at least one embodiment, computer system 900 is a single processor desktop or server system, but in another embodiment computer system 900 may be a multiprocessor system. In at least one embodiment, processor 902 may include, without limitation, a complex instruction set computer (“CISC”) microprocessor, a reduced instruction set computing (“RISC”) microprocessor, a very long instruction word (“VLIW”) microprocessor, a processor implementing a combination of instruction sets, or any other processor device, such as a digital signal processor, for example. In at least one embodiment, processor 902 may be coupled to a processor bus 910 that may transmit data signals between processor 902 and other components in computer system 900.

[0075] In at least one embodiment, processor 902 may include, without limitation, a Level 1 (“L1”) internal cache memory (“cache”) 904. In at least one embodiment, processor 902 may have a single internal cache or multiple levels of internal cache. In at least one embodiment, cache memory may reside external to processor 902. Other embodiments may also include a combination of both internal and external caches depending on particular implementation and needs. In at least one embodiment, register file 906 may store different types of data in various registers including, without limitation, integer registers, floating point registers, status registers, and instruction pointer register.

[0076] In at least one embodiment, execution unit 908, including, without limitation, logic to perform integer and floating point operations, also resides in processor 902. In at least one embodiment, processor 902 may also include a microcode (“ucode”) read only memory (“ROM”) that stores microcode for certain macro instructions. In at least one embodiment, execution unit 908 may include logic to handle a packed instruction set 909. In at least one embodiment, by including packed instruction set 909 in an instruction set of a general-purpose processor 902, along with associated circuitry to execute instructions, operations used by many

multimedia applications may be performed using packed data in a general-purpose processor 902. In one or more embodiments, many multimedia applications may be accelerated and executed more efficiently by using full width of a processor's data bus for performing operations on packed data, which may eliminate need to transfer smaller units of data across processor's data bus to perform one or more operations one data element at a time.

[0077] In at least one embodiment, execution unit 908 may also be used in microcontrollers, embedded processors, graphics devices, DSPs, and other types of logic circuits. In at least one embodiment, computer system 900 may include, without limitation, a memory 920. In at least one embodiment, memory 920 may be implemented as a Dynamic Random Access Memory ("DRAM") device, a Static Random Access Memory ("SRAM") device, flash memory device, or other memory device. In at least one embodiment, memory 920 may store instruction(s) 919 and/or data 921 represented by data signals that may be executed by processor 902.

[0078] In at least one embodiment, system logic chip may be coupled to processor bus 910 and memory 920. In at least one embodiment, system logic chip may include, without limitation, a memory controller hub ("MCH") 916, and processor 902 may communicate with MCH 916 via processor bus 910. In at least one embodiment, MCH 916 may provide a high bandwidth memory path 918 to memory 920 for instruction and data storage and for storage of graphics commands, data and textures. In at least one embodiment, MCH 916 may direct data signals between processor 902, memory 920, and other components in computer system 900 and to bridge data signals between processor bus 910, memory 920, and a system I/O 922. In at least one embodiment, system logic chip may provide a graphics port for coupling to a graphics controller. In at least one embodiment, MCH 916 may be coupled to memory 920 through a high bandwidth memory path 918 and graphics/video card 912 may be coupled to MCH 916 through an Accelerated Graphics Port ("AGP") interconnect 914.

[0079] In at least one embodiment, computer system 900 may use system I/O 922 that is a proprietary hub interface bus to couple MCH 916 to I/O controller hub ("ICH") 930. In at least one embodiment, ICH 930 may provide direct connections to some I/O devices via a local I/O bus. In at least one embodiment, local I/O bus may include, without limitation, a high-speed I/O bus for connecting peripherals to memory 920, chipset, and processor 902. Examples may

include, without limitation, an audio controller 929, a firmware hub (“flash BIOS”) 928, a wireless transceiver 926, a data storage 924, a legacy I/O controller 923 containing user input and keyboard interfaces 925, a serial expansion port 927, such as Universal Serial Bus (“USB”), and a network controller 934. Data storage 924 may comprise a hard disk drive, a floppy disk drive, a CD-ROM device, a flash memory device, or other mass storage device.

[0080] In at least one embodiment, FIG. 9 illustrates a system, which includes interconnected hardware devices or “chips”, whereas in other embodiments, FIG. 9 may illustrate an exemplary System on a Chip (“SoC”). In at least one embodiment, devices may be interconnected with proprietary interconnects, standardized interconnects (e.g., PCIe) or some combination thereof. In at least one embodiment, one or more components of computer system 900 are interconnected using compute express link (CXL) interconnects.

[0081] Inference and/or training logic 715 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGS. 7A and/or 7B. In at least one embodiment, inference and/or training logic 715 may be used in system FIG. 9 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0082] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

[0083] FIG. 10 is a block diagram illustrating an electronic device 1000 for utilizing a processor 1010, according to at least one embodiment. In at least one embodiment, electronic device 1000 may be, for example and without limitation, a notebook, a tower server, a rack server, a blade server, a laptop, a desktop, a tablet, a mobile device, a phone, an embedded computer, or any other suitable electronic device.

[0084] In at least one embodiment, system 1000 may include, without limitation, processor 1010 communicatively coupled to any suitable number or kind of components, peripherals, modules, or devices. In at least one embodiment, processor 1010 coupled using a bus or

interface, such as a I²C bus, a System Management Bus (“SMBus”), a Low Pin Count (LPC) bus, a Serial Peripheral Interface (“SPI”), a High Definition Audio (“HDA”) bus, a Serial Advance Technology Attachment (“SATA”) bus, a Universal Serial Bus (“USB”) (versions 1, 2, 3), or a Universal Asynchronous Receiver/Transmitter (“UART”) bus. In at least one embodiment, FIG. 10 illustrates a system, which includes interconnected hardware devices or “chips”, whereas in other embodiments, FIG. 10 may illustrate an exemplary System on a Chip (“SoC”). In at least one embodiment, devices illustrated in FIG. 10 may be interconnected with proprietary interconnects, standardized interconnects (e.g., PCIe) or some combination thereof. In at least one embodiment, one or more components of FIG. 10 are interconnected using compute express link (CXL) interconnects.

[0085] In at least one embodiment, FIG 10 may include a display 1024, a touch screen 1025, a touch pad 1030, a Near Field Communications unit (“NFC”) 1045, a sensor hub 1040, a thermal sensor 1046, an Express Chipset (“EC”) 1035, a Trusted Platform Module (“TPM”) 1038, BIOS/firmware/flash memory (“BIOS, FW Flash”) 1022, a DSP 1060, a drive 1020 such as a Solid State Disk (“SSD”) or a Hard Disk Drive (“HDD”), a wireless local area network unit (“WLAN”) 1050, a Bluetooth unit 1052, a Wireless Wide Area Network unit (“WWAN”) 1056, a Global Positioning System (GPS) 1055, a camera (“USB 3.0 camera”) 1054 such as a USB 3.0 camera, and/or a Low Power Double Data Rate (“LPDDR”) memory unit (“LPDDR3”) 1015 implemented in, for example, LPDDR3 standard. These components may each be implemented in any suitable manner.

[0086] In at least one embodiment, other components may be communicatively coupled to processor 1010 through components discussed above. In at least one embodiment, an accelerometer 1041, Ambient Light Sensor (“ALS”) 1042, compass 1043, and a gyroscope 1044 may be communicatively coupled to sensor hub 1040. In at least one embodiment, thermal sensor 1039, a fan 1037, a keyboard 1046, and a touch pad 1030 may be communicatively coupled to EC 1035. In at least one embodiment, speaker 1063, headphones 1064, and microphone (“mic”) 1065 may be communicatively coupled to an audio unit (“audio codec and class d amp”) 1062, which may in turn be communicatively coupled to DSP 1060. In at least one embodiment, audio unit 1064 may include, for example and without limitation, an audio

coder/decoder (“codec”) and a class D amplifier. In at least one embodiment, SIM card (“SIM”) 1057 may be communicatively coupled to WWAN unit 1056. In at least one embodiment, components such as WLAN unit 1050 and Bluetooth unit 1052, as well as WWAN unit 1056 may be implemented in a Next Generation Form Factor (“NGFF”).

[0087] Inference and/or training logic 715 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGs. 7a and/or 7b. In at least one embodiment, inference and/or training logic 715 may be used in system FIG. 10 for inferencing or predicting operations based, at least in part, on weight parameters calculated using neural network training operations, neural network functions and/or architectures, or neural network use cases described herein.

[0088] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

[0089] FIG. 11 is a block diagram of a processing system, according to at least one embodiment. In at least one embodiment, system 1100 includes one or more processors 1102 and one or more graphics processors 1108, and may be a single processor desktop system, a multiprocessor workstation system, or a server system having a large number of processors 1102 or processor cores 1107. In at least one embodiment, system 1100 is a processing platform incorporated within a system-on-a-chip (SoC) integrated circuit for use in mobile, handheld, or embedded devices.

[0090] In at least one embodiment, system 1100 can include, or be incorporated within a server-based gaming platform, a game console, including a game and media console, a mobile gaming console, a handheld game console, or an online game console. In at least one embodiment, system 1100 is a mobile phone, smart phone, tablet computing device or mobile Internet device. In at least one embodiment, processing system 1100 can also include, couple with, or be integrated within a wearable device, such as a smart watch wearable device, smart eyewear device, augmented reality device, or virtual reality device. In at least one embodiment,

processing system 1100 is a television or set top box device having one or more processors 1102 and a graphical interface generated by one or more graphics processors 1108.

[0091] In at least one embodiment, one or more processors 1102 each include one or more processor cores 1107 to process instructions which, when executed, perform operations for system and user software. In at least one embodiment, each of one or more processor cores 1107 is configured to process a specific instruction set 1109. In at least one embodiment, instruction set 1109 may facilitate Complex Instruction Set Computing (CISC), Reduced Instruction Set Computing (RISC), or computing via a Very Long Instruction Word (VLIW). In at least one embodiment, processor cores 1107 may each process a different instruction set 1109, which may include instructions to facilitate emulation of other instruction sets. In at least one embodiment, processor core 1107 may also include other processing devices, such a Digital Signal Processor (DSP).

[0092] In at least one embodiment, processor 1102 includes cache memory 1104. In at least one embodiment, processor 1102 can have a single internal cache or multiple levels of internal cache. In at least one embodiment, cache memory is shared among various components of processor 1102. In at least one embodiment, processor 1102 also uses an external cache (e.g., a Level-3 (L3) cache or Last Level Cache (LLC)) (not shown), which may be shared among processor cores 1107 using known cache coherency techniques. In at least one embodiment, register file 1106 is additionally included in processor 1102 which may include different types of registers for storing different types of data (e.g., integer registers, floating point registers, status registers, and an instruction pointer register). In at least one embodiment, register file 1106 may include general-purpose registers or other registers.

[0093] In at least one embodiment, one or more processor(s) 1102 are coupled with one or more interface bus(es) 1110 to transmit communication signals such as address, data, or control signals between processor 1102 and other components in system 1100. In at least one embodiment, interface bus 1110, in one embodiment, can be a processor bus, such as a version of a Direct Media Interface (DMI) bus. In at least one embodiment, interface 1110 is not limited to a DMI bus, and may include one or more Peripheral Component Interconnect buses (e.g., PCI, PCI Express), memory busses, or other types of interface busses. In at least one embodiment

processor(s) 1102 include an integrated memory controller 1116 and a platform controller hub 1130. In at least one embodiment, memory controller 1116 facilitates communication between a memory device and other components of system 1100, while platform controller hub (PCH) 1130 provides connections to I/O devices via a local I/O bus.

[0094] In at least one embodiment, memory device 1120 can be a dynamic random access memory (DRAM) device, a static random access memory (SRAM) device, flash memory device, phase-change memory device, or some other memory device having suitable performance to serve as process memory. In at least one embodiment memory device 1120 can operate as system memory for system 1100, to store data 1122 and instructions 1121 for use when one or more processors 1102 executes an application or process. In at least one embodiment, memory controller 1116 also couples with an optional external graphics processor 1112, which may communicate with one or more graphics processors 1108 in processors 1102 to perform graphics and media operations. In at least one embodiment, a display device 1111 can connect to processor(s) 1102. In at least one embodiment display device 1111 can include one or more of an internal display device, as in a mobile electronic device or a laptop device or an external display device attached via a display interface (e.g., DisplayPort, etc.). In at least one embodiment, display device 1111 can include a head mounted display (HMD) such as a stereoscopic display device for use in virtual reality (VR) applications or augmented reality (AR) applications.

[0095] In at least one embodiment, platform controller hub 1130 enables peripherals to connect to memory device 1120 and processor 1102 via a high-speed I/O bus. In at least one embodiment, I/O peripherals include, but are not limited to, an audio controller 1146, a network controller 1134, a firmware interface 1128, a wireless transceiver 1126, touch sensors 1125, a data storage device 1124 (e.g., hard disk drive, flash memory, etc.). In at least one embodiment, data storage device 1124 can connect via a storage interface (e.g., SATA) or via a peripheral bus, such as a Peripheral Component Interconnect bus (e.g., PCI, PCI Express). In at least one embodiment, touch sensors 1125 can include touch screen sensors, pressure sensors, or fingerprint sensors. In at least one embodiment, wireless transceiver 1126 can be a Wi-Fi transceiver, a Bluetooth transceiver, or a mobile network transceiver such as a 3G, 4G, or Long

Term Evolution (LTE) transceiver. In at least one embodiment, firmware interface 1128 enables communication with system firmware, and can be, for example, a unified extensible firmware interface (UEFI). In at least one embodiment, network controller 1134 can enable a network connection to a wired network. In at least one embodiment, a high-performance network controller (not shown) couples with interface bus 1110. In at least one embodiment, audio controller 1146 is a multi-channel high definition audio controller. In at least one embodiment, system 1100 includes an optional legacy I/O controller 1140 for coupling legacy (e.g., Personal System 2 (PS/2)) devices to system. In at least one embodiment, platform controller hub 1130 can also connect to one or more Universal Serial Bus (USB) controllers 1142 connect input devices, such as keyboard and mouse 1143 combinations, a camera 1144, or other USB input devices.

[0096] In at least one embodiment, an instance of memory controller 1116 and platform controller hub 1130 may be integrated into a discreet external graphics processor, such as external graphics processor 1112. In at least one embodiment, platform controller hub 1130 and/or memory controller 1116 may be external to one or more processor(s) 1102. For example, in at least one embodiment, system 1100 can include an external memory controller 1116 and platform controller hub 1130, which may be configured as a memory controller hub and peripheral controller hub within a system chipset that is in communication with processor(s) 1102.

[0097] Inference and/or training logic 715 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGS. 7A and/or 7B. In at least one embodiment portions or all of inference and/or training logic 715 may be incorporated into graphics processor 1500. For example, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs embodied in a graphics processor. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 7A or 7B. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of a graphics processor to perform one or more

machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0098] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

[0099] FIG. 12 is a block diagram of a processor 1200 having one or more processor cores 1202A-1202N, an integrated memory controller 1214, and an integrated graphics processor 1208, according to at least one embodiment. In at least one embodiment, processor 1200 can include additional cores up to and including additional core 1202N represented by dashed lined boxes. In at least one embodiment, each of processor cores 1202A-1202N includes one or more internal cache units 1204A-1204N. In at least one embodiment, each processor core also has access to one or more shared cache units 1206.

[0100] In at least one embodiment, internal cache units 1204A-1204N and shared cache units 1206 represent a cache memory hierarchy within processor 1200. In at least one embodiment, cache memory units 1204A-1204N may include at least one level of instruction and data cache within each processor core and one or more levels of shared mid-level cache, such as a Level 2 (L2), Level 3 (L3), Level 4 (L4), or other levels of cache, where a highest level of cache before external memory is classified as an LLC. In at least one embodiment, cache coherency logic maintains coherency between various cache units 1206 and 1204A-1204N.

[0101] In at least one embodiment, processor 1200 may also include a set of one or more bus controller units 1216 and a system agent core 1210. In at least one embodiment, one or more bus controller units 1216 manage a set of peripheral buses, such as one or more PCI or PCI express busses. In at least one embodiment, system agent core 1210 provides management functionality for various processor components. In at least one embodiment, system agent core 1210 includes one or more integrated memory controllers 1214 to manage access to various external memory devices (not shown).

[0102] In at least one embodiment, one or more of processor cores 1202A-1202N include support for simultaneous multi-threading. In at least one embodiment, system agent core 1210 includes components for coordinating and operating cores 1202A-1202N during multi-threaded

processing. In at least one embodiment, system agent core 1210 may additionally include a power control unit (PCU), which includes logic and components to regulate one or more power states of processor cores 1202A-1202N and graphics processor 1208.

[0103] In at least one embodiment, processor 1200 additionally includes graphics processor 1208 to execute graphics processing operations. In at least one embodiment, graphics processor 1208 couples with shared cache units 1206, and system agent core 1210, including one or more integrated memory controllers 1214. In at least one embodiment, system agent core 1210 also includes a display controller 1211 to drive graphics processor output to one or more coupled displays. In at least one embodiment, display controller 1211 may also be a separate module coupled with graphics processor 1208 via at least one interconnect, or may be integrated within graphics processor 1208.

[0104] In at least one embodiment, a ring based interconnect unit 1212 is used to couple internal components of processor 1200. In at least one embodiment, an alternative interconnect unit may be used, such as a point-to-point interconnect, a switched interconnect, or other techniques. In at least one embodiment, graphics processor 1208 couples with ring interconnect 1212 via an I/O link 1213.

[0105] In at least one embodiment, I/O link 1213 represents at least one of multiple varieties of I/O interconnects, including an on package I/O interconnect which facilitates communication between various processor components and a high-performance embedded memory module 1218, such as an eDRAM module. In at least one embodiment, each of processor cores 1202A-1202N and graphics processor 1208 use embedded memory modules 1218 as a shared Last Level Cache.

[0106] In at least one embodiment, processor cores 1202A-1202N are homogenous cores executing a common instruction set architecture. In at least one embodiment, processor cores 1202A-1202N are heterogeneous in terms of instruction set architecture (ISA), where one or more of processor cores 1202A-1202N execute a common instruction set, while one or more other cores of processor cores 1202A-1202N executes a subset of a common instruction set or a different instruction set. In at least one embodiment, processor cores 1202A-1202N are

heterogeneous in terms of microarchitecture, where one or more cores having a relatively higher power consumption couple with one or more power cores having a lower power consumption. In at least one embodiment, processor 1200 can be implemented on one or more chips or as an SoC integrated circuit.

[0107] Inference and/or training logic 715 are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic 715 are provided below in conjunction with FIGs. 7a and/or 7b. In at least one embodiment portions or all of inference and/or training logic 715 may be incorporated into processor 1200. For example, in at least one embodiment, training and/or inferencing techniques described herein may use one or more of ALUs embodied in graphics processor 1512, graphics core(s) 1202A-1202N, or other components in FIG. 12. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIGS. 7A or 7B. In at least one embodiment, weight parameters may be stored in on-chip or off-chip memory and/or registers (shown or not shown) that configure ALUs of graphics processor 1200 to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0108] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

VIRTUALIZED COMPUTING PLATFORM

[0109] FIG. 13 is an example data flow diagram for a process 1300 of generating and deploying an image processing and inferencing pipeline, in accordance with at least one embodiment. In at least one embodiment, process 1300 may be deployed for use with imaging devices, processing devices, and/or other device types at one or more facilities 1302. Process 1300 may be executed within a training system 1304 and/or a deployment system 1306. In at least one embodiment, training system 1304 may be used to perform training, deployment, and implementation of machine learning models (e.g., neural networks, object detection algorithms, computer vision algorithms, etc.) for use in deployment system 1306. In at least one embodiment, deployment system 1306 may be configured to offload processing and compute

resources among a distributed computing environment to reduce infrastructure requirements at facility 1302. In at least one embodiment, one or more applications in a pipeline may use or call upon services (e.g., inference, visualization, compute, AI, etc.) of deployment system 1306 during execution of applications.

[0110] In at least one embodiment, some of applications used in advanced processing and inferencing pipelines may use machine learning models or other AI to perform one or more processing steps. In at least one embodiment, machine learning models may be trained at facility 1302 using data 1308 (such as imaging data) generated at facility 1302 (and stored on one or more picture archiving and communication system (PACS) servers at facility 1302), may be trained using imaging or sequencing data 1308 from another facility(ies), or a combination thereof. In at least one embodiment, training system 1304 may be used to provide applications, services, and/or other resources for generating working, deployable machine learning models for deployment system 1306.

[0111] In at least one embodiment, model registry 1324 may be backed by object storage that may support versioning and object metadata. In at least one embodiment, object storage may be accessible through, for example, a cloud storage (e.g., cloud 1426 of FIG. 14) compatible application programming interface (API) from within a cloud platform. In at least one embodiment, machine learning models within model registry 1324 may be uploaded, listed, modified, or deleted by developers or partners of a system interacting with an API. In at least one embodiment, an API may provide access to methods that allow users with appropriate credentials to associate models with applications, such that models may be executed as part of execution of containerized instantiations of applications.

[0112] In at least one embodiment, training pipeline 1404 (FIG. 14) may include a scenario where facility 1302 is training their own machine learning model, or has an existing machine learning model that needs to be optimized or updated. In at least one embodiment, imaging data 1308 generated by imaging device(s), sequencing devices, and/or other device types may be received. In at least one embodiment, once imaging data 1308 is received, AI-assisted annotation 1310 may be used to aid in generating annotations corresponding to imaging data 1308 to be used as ground truth data for a machine learning model. In at least one embodiment,

AI-assisted annotation 1310 may include one or more machine learning models (e.g., convolutional neural networks (CNNs)) that may be trained to generate annotations corresponding to certain types of imaging data 1308 (e.g., from certain devices). In at least one embodiment, AI-assisted annotations 1310 may then be used directly, or may be adjusted or fine-tuned using an annotation tool to generate ground truth data. In at least one embodiment, AI-assisted annotations 1310, labeled clinic data 1312, or a combination thereof may be used as ground truth data for training a machine learning model. In at least one embodiment, a trained machine learning model may be referred to as output model 1316, and may be used by deployment system 1306, as described herein.

[0113] In at least one embodiment, training pipeline 1404 (FIG. 14) may include a scenario where facility 1302 needs a machine learning model for use in performing one or more processing tasks for one or more applications in deployment system 1306, but facility 1302 may not currently have such a machine learning model (or may not have a model that is optimized, efficient, or effective for such purposes). In at least one embodiment, an existing machine learning model may be selected from a model registry 1324. In at least one embodiment, model registry 1324 may include machine learning models trained to perform a variety of different inference tasks on imaging data. In at least one embodiment, machine learning models in model registry 1324 may have been trained on imaging data from different facilities than facility 1302 (e.g., facilities remotely located). In at least one embodiment, machine learning models may have been trained on imaging data from one location, two locations, or any number of locations. In at least one embodiment, when being trained on imaging data from a specific location, training may take place at that location, or at least in a manner that protects confidentiality of imaging data or restricts imaging data from being transferred off-premises. In at least one embodiment, once a model is trained – or partially trained – at one location, a machine learning model may be added to model registry 1324. In at least one embodiment, a machine learning model may then be retrained, or updated, at any number of other facilities, and a retrained or updated model may be made available in model registry 1324. In at least one embodiment, a machine learning model may then be selected from model registry 1324 – and referred to as

output model 1316 – and may be used in deployment system 1306 to perform one or more processing tasks for one or more applications of a deployment system.

[0114] In at least one embodiment, training pipeline 1404 (FIG. 14), a scenario may include facility 1302 requiring a machine learning model for use in performing one or more processing tasks for one or more applications in deployment system 1306, but facility 1302 may not currently have such a machine learning model (or may not have a model that is optimized, efficient, or effective for such purposes). In at least one embodiment, a machine learning model selected from model registry 1324 may not be fine-tuned or optimized for imaging data 1308 generated at facility 1302 because of differences in populations, robustness of training data used to train a machine learning model, diversity in anomalies of training data, and/or other issues with training data. In at least one embodiment, AI-assisted annotation 1310 may be used to aid in generating annotations corresponding to imaging data 1308 to be used as ground truth data for retraining or updating a machine learning model. In at least one embodiment, labeled data 1312 may be used as ground truth data for training a machine learning model. In at least one embodiment, retraining or updating a machine learning model may be referred to as model training 1314. In at least one embodiment, model training 1314 – e.g., AI-assisted annotations 1310, labeled clinic data 1312, or a combination thereof – may be used as ground truth data for retraining or updating a machine learning model. In at least one embodiment, a trained machine learning model may be referred to as output model 1316, and may be used by deployment system 1306, as described herein.

[0115] In at least one embodiment, deployment system 1306 may include software 1318, services 1320, hardware 1322, and/or other components, features, and functionality. In at least one embodiment, deployment system 1306 may include a software “stack,” such that software 1318 may be built on top of services 1320 and may use services 1320 to perform some or all of processing tasks, and services 1320 and software 1318 may be built on top of hardware 1322 and use hardware 1322 to execute processing, storage, and/or other compute tasks of deployment system 1306. In at least one embodiment, software 1318 may include any number of different containers, where each container may execute an instantiation of an application. In at least one embodiment, each application may perform one or more processing tasks in an advanced

processing and inferencing pipeline (e.g., inferencing, object detection, feature detection, segmentation, image enhancement, calibration, etc.). In at least one embodiment, an advanced processing and inferencing pipeline may be defined based on selections of different containers that are desired or required for processing imaging data 1308, in addition to containers that receive and configure imaging data for use by each container and/or for use by facility 1302 after processing through a pipeline (e.g., to convert outputs back to a usable data type). In at least one embodiment, a combination of containers within software 1318 (e.g., that make up a pipeline) may be referred to as a virtual instrument (as described in more detail herein), and a virtual instrument may leverage services 1320 and hardware 1322 to execute some or all processing tasks of applications instantiated in containers.

[0116] In at least one embodiment, a data processing pipeline may receive input data (e.g., imaging data 1308) in a specific format in response to an inference request (e.g., a request from a user of deployment system 1306). In at least one embodiment, input data may be representative of one or more images, video, and/or other data representations generated by one or more imaging devices. In at least one embodiment, data may undergo pre-processing as part of data processing pipeline to prepare data for processing by one or more applications. In at least one embodiment, post-processing may be performed on an output of one or more inferencing tasks or other processing tasks of a pipeline to prepare an output data for a next application and/or to prepare output data for transmission and/or use by a user (e.g., as a response to an inference request). In at least one embodiment, inferencing tasks may be performed by one or more machine learning models, such as trained or deployed neural networks, which may include output models 1316 of training system 1304.

[0117] In at least one embodiment, tasks of data processing pipeline may be encapsulated in a container(s) that each represents a discrete, fully functional instantiation of an application and virtualized computing environment that is able to reference machine learning models. In at least one embodiment, containers or applications may be published into a private (e.g., limited access) area of a container registry (described in more detail herein), and trained or deployed models may be stored in model registry 1324 and associated with one or more applications. In at least one embodiment, images of applications (e.g., container images) may be available in a container

registry, and once selected by a user from a container registry for deployment in a pipeline, an image may be used to generate a container for an instantiation of an application for use by a user's system.

[0118] In at least one embodiment, developers (e.g., software developers, clinicians, doctors, etc.) may develop, publish, and store applications (e.g., as containers) for performing image processing and/or inferencing on supplied data. In at least one embodiment, development, publishing, and/or storing may be performed using a software development kit (SDK) associated with a system (e.g., to ensure that an application and/or container developed is compliant with or compatible with a system). In at least one embodiment, an application that is developed may be tested locally (e.g., at a first facility, on data from a first facility) with an SDK which may support at least some of services 1320 as a system (e.g., system 1400 of FIG. 14). In at least one embodiment, because DICOM objects may contain anywhere from one to hundreds of images or other data types, and due to a variation in data, a developer may be responsible for managing (e.g., setting constructs for, building pre-processing into an application, etc.) extraction and preparation of incoming data. In at least one embodiment, once validated by system 1400 (e.g., for accuracy), an application may be available in a container registry for selection and/or implementation by a user to perform one or more processing tasks with respect to data at a facility (e.g., a second facility) of a user.

[0119] In at least one embodiment, developers may then share applications or containers through a network for access and use by users of a system (e.g., system 1400 of FIG. 14). In at least one embodiment, completed and validated applications or containers may be stored in a container registry and associated machine learning models may be stored in model registry 1324. In at least one embodiment, a requesting entity – who provides an inference or image processing request – may browse a container registry and/or model registry 1324 for an application, container, dataset, machine learning model, etc., select a desired combination of elements for inclusion in data processing pipeline, and submit an imaging processing request. In at least one embodiment, a request may include input data (and associated patient data, in some examples) that is necessary to perform a request, and/or may include a selection of application(s) and/or machine learning models to be executed in processing a request. In at least one embodiment, a

request may then be passed to one or more components of deployment system 1306 (e.g., a cloud) to perform processing of data processing pipeline. In at least one embodiment, processing by deployment system 1306 may include referencing selected elements (e.g., applications, containers, models, etc.) from a container registry and/or model registry 1324. In at least one embodiment, once results are generated by a pipeline, results may be returned to a user for reference (e.g., for viewing in a viewing application suite executing on a local, on-premises workstation or terminal).

[0120] In at least one embodiment, to aid in processing or execution of applications or containers in pipelines, services 1320 may be leveraged. In at least one embodiment, services 1320 may include compute services, artificial intelligence (AI) services, visualization services, and/or other service types. In at least one embodiment, services 1320 may provide functionality that is common to one or more applications in software 1318, so functionality may be abstracted to a service that may be called upon or leveraged by applications. In at least one embodiment, functionality provided by services 1320 may run dynamically and more efficiently, while also scaling well by allowing applications to process data in parallel (e.g., using a parallel computing platform 1430 (FIG. 14)). In at least one embodiment, rather than each application that shares a same functionality offered by a service 1320 being required to have a respective instance of service 1320, service 1320 may be shared between and among various applications. In at least one embodiment, services may include an inference server or engine that may be used for executing detection or segmentation tasks, as non-limiting examples. In at least one embodiment, a model training service may be included that may provide machine learning model training and/or retraining capabilities. In at least one embodiment, a data augmentation service may further be included that may provide GPU accelerated data (e.g., DICOM, RIS, CIS, REST compliant, RPC, raw, etc.) extraction, resizing, scaling, and/or other augmentation. In at least one embodiment, a visualization service may be used that may add image rendering effects – such as ray-tracing, rasterization, denoising, sharpening, etc. – to add realism to two-dimensional (2D) and/or three-dimensional (3D) models. In at least one embodiment, virtual instrument services may be included that provide for beam-forming, segmentation, inferencing, imaging, and/or support for other applications within pipelines of virtual instruments.

[0121] In at least one embodiment, where a service 1320 includes an AI service (e.g., an inference service), one or more machine learning models may be executed by calling upon (e.g., as an API call) an inference service (e.g., an inference server) to execute machine learning model(s), or processing thereof, as part of application execution. In at least one embodiment, where another application includes one or more machine learning models for segmentation tasks, an application may call upon an inference service to execute machine learning models for performing one or more of processing operations associated with segmentation tasks. In at least one embodiment, software 1318 implementing advanced processing and inferencing pipeline that includes segmentation application and anomaly detection application may be streamlined because each application may call upon a same inference service to perform one or more inferencing tasks.

[0122] In at least one embodiment, hardware 1322 may include GPUs, CPUs, graphics cards, an AI/deep learning system (e.g., an AI supercomputer, such as NVIDIA's DGX), a cloud platform, or a combination thereof. In at least one embodiment, different types of hardware 1322 may be used to provide efficient, purpose-built support for software 1318 and services 1320 in deployment system 1306. In at least one embodiment, use of GPU processing may be implemented for processing locally (e.g., at facility 1302), within an AI/deep learning system, in a cloud system, and/or in other processing components of deployment system 1306 to improve efficiency, accuracy, and efficacy of image processing and generation. In at least one embodiment, software 1318 and/or services 1320 may be optimized for GPU processing with respect to deep learning, machine learning, and/or high-performance computing, as non-limiting examples. In at least one embodiment, at least some of computing environment of deployment system 1306 and/or training system 1304 may be executed in a datacenter one or more supercomputers or high performance computing systems, with GPU optimized software (e.g., hardware and software combination of NVIDIA's DGX System). In at least one embodiment, hardware 1322 may include any number of GPUs that may be called upon to perform processing of data in parallel, as described herein. In at least one embodiment, cloud platform may further include GPU processing for GPU-optimized execution of deep learning tasks, machine learning tasks, or other computing tasks. In at least one embodiment, cloud platform (e.g., NVIDIA's

NGC) may be executed using an AI/deep learning supercomputer(s) and/or GPU-optimized software (e.g., as provided on NVIDIA's DGX Systems) as a hardware abstraction and scaling platform. In at least one embodiment, cloud platform may integrate an application container clustering system or orchestration system (e.g., KUBERNETES) on multiple GPUs to enable seamless scaling and load balancing.

[0123] FIG. 14 is a system diagram for an example system 1400 for generating and deploying an imaging deployment pipeline, in accordance with at least one embodiment. In at least one embodiment, system 1400 may be used to implement process 1300 of FIG. 13 and/or other processes including advanced processing and inferencing pipelines. In at least one embodiment, system 1400 may include training system 1304 and deployment system 1306. In at least one embodiment, training system 1304 and deployment system 1306 may be implemented using software 1318, services 1320, and/or hardware 1322, as described herein.

[0124] In at least one embodiment, system 1400 (e.g., training system 1304 and/or deployment system 1306) may be implemented in a cloud computing environment (e.g., using cloud 1426). In at least one embodiment, system 1400 may be implemented locally with respect to a healthcare services facility, or as a combination of both cloud and local computing resources. In at least one embodiment, access to APIs in cloud 1426 may be restricted to authorized users through enacted security measures or protocols. In at least one embodiment, a security protocol may include web tokens that may be signed by an authentication (e.g., AuthN, AuthZ, Gluecon, etc.) service and may carry appropriate authorization. In at least one embodiment, APIs of virtual instruments (described herein), or other instantiations of system 1400, may be restricted to a set of public IPs that have been vetted or authorized for interaction.

[0125] In at least one embodiment, various components of system 1400 may communicate between and among one another using any of a variety of different network types, including but not limited to local area networks (LANs) and/or wide area networks (WANs) via wired and/or wireless communication protocols. In at least one embodiment, communication between facilities and components of system 1400 (e.g., for transmitting inference requests, for receiving results of inference requests, etc.) may be communicated over data bus(es), wireless data protocols (Wi-Fi), wired data protocols (e.g., Ethernet), etc.

[0126] In at least one embodiment, training system 1304 may execute training pipelines 1404, similar to those described herein with respect to FIG. 13. In at least one embodiment, where one or more machine learning models are to be used in deployment pipelines 1410 by deployment system 1306, training pipelines 1404 may be used to train or retrain one or more (e.g. pre-trained) models, and/or implement one or more of pre-trained models 1406 (e.g., without a need for retraining or updating). In at least one embodiment, as a result of training pipelines 1404, output model(s) 1316 may be generated. In at least one embodiment, training pipelines 1404 may include any number of processing steps, such as but not limited to imaging data (or other input data) conversion or adaption. In at least one embodiment, for different machine learning models used by deployment system 1306, different training pipelines 1404 may be used. In at least one embodiment, training pipeline 1404 similar to a first example described with respect to FIG. 13 may be used for a first machine learning model, training pipeline 1404 similar to a second example described with respect to FIG. 13 may be used for a second machine learning model, and training pipeline 1404 similar to a third example described with respect to FIG. 13 may be used for a third machine learning model. In at least one embodiment, any combination of tasks within training system 1304 may be used depending on what is required for each respective machine learning model. In at least one embodiment, one or more of machine learning models may already be trained and ready for deployment so machine learning models may not undergo any processing by training system 1304, and may be implemented by deployment system 1306.

[0127] In at least one embodiment, output model(s) 1316 and/or pre-trained model(s) 1406 may include any types of machine learning models depending on implementation or embodiment. In at least one embodiment, and without limitation, machine learning models used by system 1400 may include machine learning model(s) using linear regression, logistic regression, decision trees, support vector machines (SVM), Naïve Bayes, k-nearest neighbor (Knn), K means clustering, random forest, dimensionality reduction algorithms, gradient boosting algorithms, neural networks (e.g., auto-encoders, convolutional, recurrent, perceptrons, Long/Short Term Memory (LSTM), Hopfield, Boltzmann, deep belief, deconvolutional, generative adversarial, liquid state machine, etc.), and/or other types of machine learning models.

[0128] In at least one embodiment, training pipelines 1404 may include AI-assisted annotation, as described in more detail herein with respect to at least FIG. 15B. In at least one embodiment, labeled data 1312 (e.g., traditional annotation) may be generated by any number of techniques. In at least one embodiment, labels or other annotations may be generated within a drawing program (e.g., an annotation program), a computer aided design (CAD) program, a labeling program, another type of program suitable for generating annotations or labels for ground truth, and/or may be hand drawn, in some examples. In at least one embodiment, ground truth data may be synthetically produced (e.g., generated from computer models or renderings), real produced (e.g., designed and produced from real-world data), machine-automated (e.g., using feature analysis and learning to extract features from data and then generate labels), human annotated (e.g., labeler, or annotation expert, defines location of labels), and/or a combination thereof. In at least one embodiment, for each instance of imaging data 1308 (or other data type used by machine learning models), there may be corresponding ground truth data generated by training system 1304. In at least one embodiment, AI-assisted annotation may be performed as part of deployment pipelines 1410; either in addition to, or in lieu of AI-assisted annotation included in training pipelines 1404. In at least one embodiment, system 1400 may include a multi-layer platform that may include a software layer (e.g., software 1318) of diagnostic applications (or other application types) that may perform one or more medical imaging and diagnostic functions. In at least one embodiment, system 1400 may be communicatively coupled to (e.g., via encrypted links) PACS server networks of one or more facilities. In at least one embodiment, system 1400 may be configured to access and referenced data from PACS servers to perform operations, such as training machine learning models, deploying machine learning models, image processing, inferencing, and/or other operations.

[0129] In at least one embodiment, a software layer may be implemented as a secure, encrypted, and/or authenticated API through which applications or containers may be invoked (e.g., called) from an external environment(s) (e.g., facility 1302). In at least one embodiment, applications may then call or execute one or more services 1320 for performing compute, AI, or visualization tasks associated with respective applications, and software 1318 and/or services

1320 may leverage hardware 1322 to perform processing tasks in an effective and efficient manner.

[0130] In at least one embodiment, deployment system 1306 may execute deployment pipelines 1410. In at least one embodiment, deployment pipelines 1410 may include any number of applications that may be sequentially, non-sequentially, or otherwise applied to imaging data (and/or other data types) generated by imaging devices, sequencing devices, genomics devices, etc. – including AI-assisted annotation, as described above. In at least one embodiment, as described herein, a deployment pipeline 1410 for an individual device may be referred to as a virtual instrument for a device (e.g., a virtual ultrasound instrument, a virtual CT scan instrument, a virtual sequencing instrument, etc.). In at least one embodiment, for a single device, there may be more than one deployment pipeline 1410 depending on information desired from data generated by a device. In at least one embodiment, where detections of anomalies are desired from an MRI machine, there may be a first deployment pipeline 1410, and where image enhancement is desired from output of an MRI machine, there may be a second deployment pipeline 1410.

[0131] In at least one embodiment, an image generation application may include a processing task that includes use of a machine learning model. In at least one embodiment, a user may desire to use their own machine learning model, or to select a machine learning model from model registry 1324. In at least one embodiment, a user may implement their own machine learning model or select a machine learning model for inclusion in an application for performing a processing task. In at least one embodiment, applications may be selectable and customizable, and by defining constructs of applications, deployment and implementation of applications for a particular user are presented as a more seamless user experience. In at least one embodiment, by leveraging other features of system 1400 – such as services 1320 and hardware 1322 – deployment pipelines 1410 may be even more user friendly, provide for easier integration, and produce more accurate, efficient, and timely results.

[0132] In at least one embodiment, deployment system 1306 may include a user interface 1414 (e.g., a graphical user interface, a web interface, etc.) that may be used to select applications for inclusion in deployment pipeline(s) 1410, arrange applications, modify or change applications or

parameters or constructs thereof, use and interact with deployment pipeline(s) 1410 during set-up and/or deployment, and/or to otherwise interact with deployment system 1306. In at least one embodiment, although not illustrated with respect to training system 1304, user interface 1414 (or a different user interface) may be used for selecting models for use in deployment system 1306, for selecting models for training, or retraining, in training system 1304, and/or for otherwise interacting with training system 1304.

[0133] In at least one embodiment, pipeline manager 1412 may be used, in addition to an application orchestration system 1428, to manage interaction between applications or containers of deployment pipeline(s) 1410 and services 1320 and/or hardware 1322. In at least one embodiment, pipeline manager 1412 may be configured to facilitate interactions from application to application, from application to service 1320, and/or from application or service to hardware 1322. In at least one embodiment, although illustrated as included in software 1318, this is not intended to be limiting, and in some examples (e.g., as illustrated in FIG. 12cc) pipeline manager 1412 may be included in services 1320. In at least one embodiment, application orchestration system 1428 (e.g., Kubernetes, DOCKER, etc.) may include a container orchestration system that may group applications into containers as logical units for coordination, management, scaling, and deployment. In at least one embodiment, by associating applications from deployment pipeline(s) 1410 (e.g., a reconstruction application, a segmentation application, etc.) with individual containers, each application may execute in a self-contained environment (e.g., at a kernel level) to increase speed and efficiency.

[0134] In at least one embodiment, each application and/or container (or image thereof) may be individually developed, modified, and deployed (e.g., a first user or developer may develop, modify, and deploy a first application and a second user or developer may develop, modify, and deploy a second application separate from a first user or developer), which may allow for focus on, and attention to, a task of a single application and/or container(s) without being hindered by tasks of another application(s) or container(s). In at least one embodiment, communication, and cooperation between different containers or applications may be aided by pipeline manager 1412 and application orchestration system 1428. In at least one embodiment, so long as an expected input and/or output of each container or application is known by a system (e.g., based on

constructs of applications or containers), application orchestration system 1428 and/or pipeline manager 1412 may facilitate communication among and between, and sharing of resources among and between, each of applications or containers. In at least one embodiment, because one or more of applications or containers in deployment pipeline(s) 1410 may share same services and resources, application orchestration system 1428 may orchestrate, load balance, and determine sharing of services or resources between and among various applications or containers. In at least one embodiment, a scheduler may be used to track resource requirements of applications or containers, current usage or planned usage of these resources, and resource availability. In at least one embodiment, a scheduler may thus allocate resources to different applications and distribute resources between and among applications in view of requirements and availability of a system. In some examples, a scheduler (and/or other component of application orchestration system 1428) may determine resource availability and distribution based on constraints imposed on a system (e.g., user constraints), such as quality of service (QoS), urgency of need for data outputs (e.g., to determine whether to execute real-time processing or delayed processing), etc.

[0135] In at least one embodiment, services 1320 leveraged by and shared by applications or containers in deployment system 1306 may include compute services 1416, AI services 1418, visualization services 1420, and/or other service types. In at least one embodiment, applications may call (e.g., execute) one or more of services 1320 to perform processing operations for an application. In at least one embodiment, compute services 1416 may be leveraged by applications to perform super-computing or other high-performance computing (HPC) tasks. In at least one embodiment, compute service(s) 1416 may be leveraged to perform parallel processing (e.g., using a parallel computing platform 1430) for processing data through one or more of applications and/or one or more tasks of a single application, substantially simultaneously. In at least one embodiment, parallel computing platform 1430 (e.g., NVIDIA's CUDA) may enable general purpose computing on GPUs (GPGPU) (e.g., GPUs 1422). In at least one embodiment, a software layer of parallel computing platform 1430 may provide access to virtual instruction sets and parallel computational elements of GPUs, for execution of compute kernels. In at least one embodiment, parallel computing platform 1430 may include memory

and, in some embodiments, a memory may be shared between and among multiple containers, and/or between and among different processing tasks within a single container. In at least one embodiment, inter-process communication (IPC) calls may be generated for multiple containers and/or for multiple processes within a container to use same data from a shared segment of memory of parallel computing platform 1430 (e.g., where multiple different stages of an application or multiple applications are processing same information). In at least one embodiment, rather than making a copy of data and moving data to different locations in memory (e.g., a read/write operation), same data in same location of a memory may be used for any number of processing tasks (e.g., at a same time, at different times, etc.). In at least one embodiment, as data is used to generate new data as a result of processing, this information of a new location of data may be stored and shared between various applications. In at least one embodiment, location of data and a location of updated or modified data may be part of a definition of how a payload is understood within containers.

[0136] In at least one embodiment, AI services 1418 may be leveraged to perform inferencing services for executing machine learning model(s) associated with applications (e.g., tasked with performing one or more processing tasks of an application). In at least one embodiment, AI services 1418 may leverage AI system 1424 to execute machine learning model(s) (e.g., neural networks, such as CNNs) for segmentation, reconstruction, object detection, feature detection, classification, and/or other inferencing tasks. In at least one embodiment, applications of deployment pipeline(s) 1410 may use one or more of output models 1316 from training system 1304 and/or other models of applications to perform inference on imaging data. In at least one embodiment, two or more examples of inferencing using application orchestration system 1428 (e.g., a scheduler) may be available. In at least one embodiment, a first category may include a high priority/low latency path that may achieve higher service level agreements, such as for performing inference on urgent requests during an emergency, or for a radiologist during diagnosis. In at least one embodiment, a second category may include a standard priority path that may be used for requests that may be non-urgent or where analysis may be performed at a later time. In at least one embodiment, application orchestration system 1428 may distribute

resources (e.g., services 1320 and/or hardware 1322) based on priority paths for different inferencing tasks of AI services 1418.

[0137] In at least one embodiment, shared storage may be mounted to AI services 1418 within system 1400. In at least one embodiment, shared storage may operate as a cache (or other storage device type) and may be used to process inference requests from applications. In at least one embodiment, when an inference request is submitted, a request may be received by a set of API instances of deployment system 1306, and one or more instances may be selected (e.g., for best fit, for load balancing, etc.) to process a request. In at least one embodiment, to process a request, a request may be entered into a database, a machine learning model may be located from model registry 1324 if not already in a cache, a validation step may ensure appropriate machine learning model is loaded into a cache (e.g., shared storage), and/or a copy of a model may be saved to a cache. In at least one embodiment, a scheduler (e.g., of pipeline manager 1412) may be used to launch an application that is referenced in a request if an application is not already running or if there are not enough instances of an application. In at least one embodiment, if an inference server is not already launched to execute a model, an inference server may be launched. Any number of inference servers may be launched per model. In at least one embodiment, in a pull model, in which inference servers are clustered, models may be cached whenever load balancing is advantageous. In at least one embodiment, inference servers may be statically loaded in corresponding, distributed servers.

[0138] In at least one embodiment, inferencing may be performed using an inference server that runs in a container. In at least one embodiment, an instance of an inference server may be associated with a model (and optionally a plurality of versions of a model). In at least one embodiment, if an instance of an inference server does not exist when a request to perform inference on a model is received, a new instance may be loaded. In at least one embodiment, when starting an inference server, a model may be passed to an inference server such that a same container may be used to serve different models so long as inference server is running as a different instance.

[0139] In at least one embodiment, during application execution, an inference request for a given application may be received, and a container (e.g., hosting an instance of an inference

server) may be loaded (if not already), and a start procedure may be called. In at least one embodiment, pre-processing logic in a container may load, decode, and/or perform any additional pre-processing on incoming data (e.g., using a CPU(s) and/or GPU(s)). In at least one embodiment, once data is prepared for inference, a container may perform inference as necessary on data. In at least one embodiment, this may include a single inference call on one image (e.g., a hand X-ray), or may require inference on hundreds of images (e.g., a chest CT). In at least one embodiment, an application may summarize results before completing, which may include, without limitation, a single confidence score, pixel level-segmentation, voxel-level segmentation, generating a visualization, or generating text to summarize findings. In at least one embodiment, different models or applications may be assigned different priorities. For example, some models may have a real-time ($TAT < 1$ min) priority while others may have lower priority (e.g., $TAT < 10$ min). In at least one embodiment, model execution times may be measured from requesting institution or entity and may include partner network traversal time, as well as execution on an inference service.

[0140] In at least one embodiment, transfer of requests between services 1320 and inference applications may be hidden behind a software development kit (SDK), and robust transport may be provide through a queue. In at least one embodiment, a request will be placed in a queue via an API for an individual application/tenant ID combination and an SDK will pull a request from a queue and give a request to an application. In at least one embodiment, a name of a queue may be provided in an environment from where an SDK will pick it up. In at least one embodiment, asynchronous communication through a queue may be useful as it may allow any instance of an application to pick up work as it becomes available. Results may be transferred back through a queue, to ensure no data is lost. In at least one embodiment, queues may also provide an ability to segment work, as highest priority work may go to a queue with most instances of an application connected to it, while lowest priority work may go to a queue with a single instance connected to it that processes tasks in an order received. In at least one embodiment, an application may run on a GPU-accelerated instance generated in cloud 1426, and an inference service may perform inferencing on a GPU.

[0141] In at least one embodiment, visualization services 1420 may be leveraged to generate visualizations for viewing outputs of applications and/or deployment pipeline(s) 1410. In at least one embodiment, GPUs 1422 may be leveraged by visualization services 1420 to generate visualizations. In at least one embodiment, rendering effects, such as ray-tracing, may be implemented by visualization services 1420 to generate higher quality visualizations. In at least one embodiment, visualizations may include, without limitation, 2D image renderings, 3D volume renderings, 3D volume reconstruction, 2D tomographic slices, virtual reality displays, augmented reality displays, etc. In at least one embodiment, virtualized environments may be used to generate a virtual interactive display or environment (e.g., a virtual environment) for interaction by users of a system (e.g., doctors, nurses, radiologists, etc.). In at least one embodiment, visualization services 1420 may include an internal visualizer, cinematics, and/or other rendering or image processing capabilities or functionality (e.g., ray tracing, rasterization, internal optics, etc.).

[0142] In at least one embodiment, hardware 1322 may include GPUs 1422, AI system 1424, cloud 1426, and/or any other hardware used for executing training system 1304 and/or deployment system 1306. In at least one embodiment, GPUs 1422 (e.g., NVIDIA's TESLA and/or QUADRO GPUs) may include any number of GPUs that may be used for executing processing tasks of compute services 1416, AI services 1418, visualization services 1420, other services, and/or any of features or functionality of software 1318. For example, with respect to AI services 1418, GPUs 1422 may be used to perform pre-processing on imaging data (or other data types used by machine learning models), post-processing on outputs of machine learning models, and/or to perform inferencing (e.g., to execute machine learning models). In at least one embodiment, cloud 1426, AI system 1424, and/or other components of system 1400 may use GPUs 1422. In at least one embodiment, cloud 1426 may include a GPU-optimized platform for deep learning tasks. In at least one embodiment, AI system 1424 may use GPUs, and cloud 1426 – or at least a portion tasked with deep learning or inferencing – may be executed using one or more AI systems 1424. As such, although hardware 1322 is illustrated as discrete components, this is not intended to be limiting, and any components of hardware 1322 may be combined with, or leveraged by, any other components of hardware 1322.

[0143] In at least one embodiment, AI system 1424 may include a purpose-built computing system (e.g., a super-computer or an HPC) configured for inferencing, deep learning, machine learning, and/or other artificial intelligence tasks. In at least one embodiment, AI system 1424 (e.g., NVIDIA's DGX) may include GPU-optimized software (e.g., a software stack) that may be executed using a plurality of GPUs 1422, in addition to CPUs, RAM, storage, and/or other components, features, or functionality. In at least one embodiment, one or more AI systems 1424 may be implemented in cloud 1426 (e.g., in a data center) for performing some or all of AI-based processing tasks of system 1400.

[0144] In at least one embodiment, cloud 1426 may include a GPU-accelerated infrastructure (e.g., NVIDIA's NGC) that may provide a GPU-optimized platform for executing processing tasks of system 1400. In at least one embodiment, cloud 1426 may include an AI system(s) 1424 for performing one or more of AI-based tasks of system 1400 (e.g., as a hardware abstraction and scaling platform). In at least one embodiment, cloud 1426 may integrate with application orchestration system 1428 leveraging multiple GPUs to enable seamless scaling and load balancing between and among applications and services 1320. In at least one embodiment, cloud 1426 may be tasked with executing at least some of services 1320 of system 1400, including compute services 1416, AI services 1418, and/or visualization services 1420, as described herein. In at least one embodiment, cloud 1426 may perform small and large batch inference (e.g., executing NVIDIA's TENSOR RT), provide an accelerated parallel computing API and platform 1430 (e.g., NVIDIA's CUDA), execute application orchestration system 1428 (e.g., KUBERNETES), provide a graphics rendering API and platform (e.g., for ray-tracing, 2D graphics, 3D graphics, and/or other rendering techniques to produce higher quality cinematics), and/or may provide other functionality for system 1400.

[0145] FIG. 15A illustrates a data flow diagram for a process 1500 to train, retrain, or update a machine learning model, in accordance with at least one embodiment. In at least one embodiment, process 1500 may be executed using, as a non-limiting example, system 1400 of FIG. 14. In at least one embodiment, process 1500 may leverage services 1320 and/or hardware 1322 of system 1400, as described herein. In at least one embodiment, refined models 1512

generated by process 1500 may be executed by deployment system 1306 for one or more containerized applications in deployment pipelines 1410.

[0146] In at least one embodiment, model training 1314 may include retraining or updating an initial model 1504 (e.g., a pre-trained model) using new training data (e.g., new input data, such as customer dataset 1506, and/or new ground truth data associated with input data). In at least one embodiment, to retrain, or update, initial model 1504, output or loss layer(s) of initial model 1504 may be reset, or deleted, and/or replaced with an updated or new output or loss layer(s). In at least one embodiment, initial model 1504 may have previously fine-tuned parameters (e.g., weights and/or biases) that remain from prior training, so training or retraining 1314 may not take as long or require as much processing as training a model from scratch. In at least one embodiment, during model training 1314, by having reset or replaced output or loss layer(s) of initial model 1504, parameters may be updated and re-tuned for a new data set based on loss calculations associated with accuracy of output or loss layer(s) at generating predictions on new, customer dataset 1506 (e.g., image data 1308 of FIG. 13).

[0147] In at least one embodiment, pre-trained models 1406 may be stored in a data store, or registry (e.g., model registry 1324 of FIG. 13). In at least one embodiment, pre-trained models 1406 may have been trained, at least in part, at one or more facilities other than a facility executing process 1500. In at least one embodiment, to protect privacy and rights of patients, subjects, or clients of different facilities, pre-trained models 1406 may have been trained, on-premise, using customer or patient data generated on-premise. In at least one embodiment, pre-trained models 1406 may be trained using cloud 1426 and/or other hardware 1322, but confidential, privacy protected patient data may not be transferred to, used by, or accessible to any components of cloud 1426 (or other off premise hardware). In at least one embodiment, where a pre-trained model 1406 is trained at using patient data from more than one facility, pre-trained model 1406 may have been individually trained for each facility prior to being trained on patient or customer data from another facility. In at least one embodiment, such as where a customer or patient data has been released of privacy concerns (e.g., by waiver, for experimental use, etc.), or where a customer or patient data is included in a public data set, a customer or

patient data from any number of facilities may be used to train pre-trained model 1406 on-premise and/or off premise, such as in a datacenter or other cloud computing infrastructure.

[0148] In at least one embodiment, when selecting applications for use in deployment pipelines 1410, a user may also select machine learning models to be used for specific applications. In at least one embodiment, a user may not have a model for use, so a user may select a pre-trained model 1406 to use with an application. In at least one embodiment, pre-trained model 1406 may not be optimized for generating accurate results on customer dataset 1506 of a facility of a user (e.g., based on patient diversity, demographics, types of medical imaging devices used, etc.). In at least one embodiment, prior to deploying pre-trained model 1406 into deployment pipeline 1410 for use with an application(s), pre-trained model 1406 may be updated, retrained, and/or fine-tuned for use at a respective facility.

[0149] In at least one embodiment, a user may select pre-trained model 1406 that is to be updated, retrained, and/or fine-tuned, and pre-trained model 1406 may be referred to as initial model 1504 for training system 1304 within process 1500. In at least one embodiment, customer dataset 1506 (e.g., imaging data, genomics data, sequencing data, or other data types generated by devices at a facility) may be used to perform model training 1314 (which may include, without limitation, transfer learning) on initial model 1504 to generate refined model 1512. In at least one embodiment, ground truth data corresponding to customer dataset 1506 may be generated by training system 1304. In at least one embodiment, ground truth data may be generated, at least in part, by clinicians, scientists, doctors, practitioners, at a facility (e.g., as labeled clinic data 1312 of FIG. 13).

[0150] In at least one embodiment, AI-assisted annotation 1310 may be used in some examples to generate ground truth data. In at least one embodiment, AI-assisted annotation 1310 (e.g., implemented using an AI-assisted annotation SDK) may leverage machine learning models (e.g., neural networks) to generate suggested or predicted ground truth data for a customer dataset. In at least one embodiment, user 1510 may use annotation tools within a user interface (a graphical user interface (GUI)) on computing device 1508.

[0151] In at least one embodiment, user 1510 may interact with a GUI via computing device 1508 to edit or fine-tune (auto)annotations. In at least one embodiment, a polygon editing feature may be used to move vertices of a polygon to more accurate or fine-tuned locations.

[0152] In at least one embodiment, once customer dataset 1506 has associated ground truth data, ground truth data (e.g., from AI-assisted annotation, manual labeling, etc.) may be used by during model training 1314 to generate refined model 1512. In at least one embodiment, customer dataset 1506 may be applied to initial model 1504 any number of times, and ground truth data may be used to update parameters of initial model 1504 until an acceptable level of accuracy is attained for refined model 1512. In at least one embodiment, once refined model 1512 is generated, refined model 1512 may be deployed within one or more deployment pipelines 1410 at a facility for performing one or more processing tasks with respect to medical imaging data.

[0153] In at least one embodiment, refined model 1512 may be uploaded to pre-trained models 1406 in model registry 1324 to be selected by another facility. In at least one embodiment, his process may be completed at any number of facilities such that refined model 1512 may be further refined on new datasets any number of times to generate a more universal model.

[0154] FIG. 15B is an example illustration of a client-server architecture 1532 to enhance annotation tools with pre-trained annotation models, in accordance with at least one embodiment. In at least one embodiment, AI-assisted annotation tools 1536 may be instantiated based on a client-server architecture 1532. In at least one embodiment, annotation tools 1536 in imaging applications may aid radiologists, for example, identify organs and abnormalities. In at least one embodiment, imaging applications may include software tools that help user 1510 to identify, as a non-limiting example, a few extreme points on a particular organ of interest in raw images 1534 (e.g., in a 3D MRI or CT scan) and receive auto-annotated results for all 2D slices of a particular organ. In at least one embodiment, results may be stored in a data store as training data 1538 and used as (for example and without limitation) ground truth data for training. In at least one embodiment, when computing device 1508 sends extreme points for AI-assisted annotation 1310, a deep learning model, for example, may receive this data as input and return inference results of a segmented organ or abnormality. In at least one embodiment, pre-

instantiated annotation tools, such as AI-Assisted Annotation Tool 1536B in FIG. 15B, may be enhanced by making API calls (e.g., API Call 1544) to a server, such as an Annotation Assistant Server 1540 that may include a set of pre-trained models 1542 stored in an annotation model registry, for example. In at least one embodiment, an annotation model registry may store pre-trained models 1542 (e.g., machine learning models, such as deep learning models) that are pre-trained to perform AI-assisted annotation on a particular organ or abnormality. These models may be further updated by using training pipelines 1404. In at least one embodiment, pre-installed annotation tools may be improved over time as new labeled clinic data 1312 is added.

[0155] Such components can be used to generate enhanced content, such as image or video content with upscaled resolution, reduced artifact presence, and visual quality enhancement.

[0156] Other variations are within spirit of present disclosure. Thus, while disclosed techniques are susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in drawings and have been described above in detail. It should be understood, however, that there is no intention to limit disclosure to specific form or forms disclosed, but on contrary, intention is to cover all modifications, alternative constructions, and equivalents falling within spirit and scope of disclosure, as defined in appended claims.

[0157] Use of terms “a” and “an” and “the” and similar referents in context of describing disclosed embodiments (especially in context of following claims) are to be construed to cover both singular and plural, unless otherwise indicated herein or clearly contradicted by context, and not as a definition of a term. Terms “comprising,” “having,” “including,” and “containing” are to be construed as open-ended terms (meaning “including, but not limited to,”) unless otherwise noted. Term “connected,” when unmodified and referring to physical connections, is to be construed as partly or wholly contained within, attached to, or joined together, even if there is something intervening. Recitation of ranges of values herein are merely intended to serve as a shorthand method of referring individually to each separate value falling within range, unless otherwise indicated herein and each separate value is incorporated into specification as if it were individually recited herein. Use of term “set” (e.g., “a set of items”) or “subset,” unless otherwise noted or contradicted by context, is to be construed as a nonempty collection comprising one or more members. Further, unless otherwise noted or contradicted by context,

term “subset” of a corresponding set does not necessarily denote a proper subset of corresponding set, but subset and corresponding set may be equal.

[0158] Conjunctive language, such as phrases of form “at least one of A, B, and C,” or “at least one of A, B and C,” unless specifically stated otherwise or otherwise clearly contradicted by context, is otherwise understood with context as used in general to present that an item, term, etc., may be either A or B or C, or any nonempty subset of set of A and B and C. For instance, in illustrative example of a set having three members, conjunctive phrases “at least one of A, B, and C” and “at least one of A, B and C” refer to any of following sets: {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, {A, B, C}. Thus, such conjunctive language is not generally intended to imply that certain embodiments require at least one of A, at least one of B, and at least one of C each to be present. In addition, unless otherwise noted or contradicted by context, term “plurality” indicates a state of being plural (e.g., “a plurality of items” indicates multiple items). A plurality is at least two items, but can be more when so indicated either explicitly or by context. Further, unless stated otherwise or otherwise clear from context, phrase “based on” means “based at least in part on” and not “based solely on.”

[0159] Operations of processes described herein can be performed in any suitable order unless otherwise indicated herein or otherwise clearly contradicted by context. In at least one embodiment, a process such as those processes described herein (or variations and/or combinations thereof) is performed under control of one or more computer systems configured with executable instructions and is implemented as code (e.g., executable instructions, one or more computer programs or one or more applications) executing collectively on one or more processors, by hardware or combinations thereof. In at least one embodiment, code is stored on a computer-readable storage medium, for example, in form of a computer program comprising a plurality of instructions executable by one or more processors. In at least one embodiment, a computer-readable storage medium is a non-transitory computer-readable storage medium that excludes transitory signals (e.g., a propagating transient electric or electromagnetic transmission) but includes non-transitory data storage circuitry (e.g., buffers, cache, and queues) within transceivers of transitory signals. In at least one embodiment, code (e.g., executable code or source code) is stored on a set of one or more non-transitory computer-readable storage media

having stored thereon executable instructions (or other memory to store executable instructions) that, when executed (i.e., as a result of being executed) by one or more processors of a computer system, cause computer system to perform operations described herein. A set of non-transitory computer-readable storage media, in at least one embodiment, comprises multiple non-transitory computer-readable storage media and one or more of individual non-transitory storage media of multiple non-transitory computer-readable storage media lack all of code while multiple non-transitory computer-readable storage media collectively store all of code. In at least one embodiment, executable instructions are executed such that different instructions are executed by different processors - for example, a non-transitory computer-readable storage medium store instructions and a main central processing unit (“CPU”) executes some of instructions while a graphics processing unit (“GPU”) executes other instructions. In at least one embodiment, different components of a computer system have separate processors and different processors execute different subsets of instructions.

[0160] Accordingly, in at least one embodiment, computer systems are configured to implement one or more services that singly or collectively perform operations of processes described herein and such computer systems are configured with applicable hardware and/or software that enable performance of operations. Further, a computer system that implements at least one embodiment of present disclosure is a single device and, in another embodiment, is a distributed computer system comprising multiple devices that operate differently such that distributed computer system performs operations described herein and such that a single device does not perform all operations.

[0161] Use of any and all examples, or exemplary language (e.g., “such as”) provided herein, is intended merely to better illuminate embodiments of disclosure and does not pose a limitation on scope of disclosure unless otherwise claimed. No language in specification should be construed as indicating any non-claimed element as essential to practice of disclosure.

[0162] All references, including publications, patent applications, and patents, cited herein are hereby incorporated by reference to same extent as if each reference were individually and specifically indicated to be incorporated by reference and were set forth in its entirety herein.

[0163] In description and claims, terms “coupled” and “connected,” along with their derivatives, may be used. It should be understood that these terms may be not intended as synonyms for each other. Rather, in particular examples, “connected” or “coupled” may be used to indicate that two or more elements are in direct or indirect physical or electrical contact with each other. “Coupled” may also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other.

[0164] Unless specifically stated otherwise, it may be appreciated that throughout specification terms such as “processing,” “computing,” “calculating,” “determining,” or like, refer to action and/or processes of a computer or computing system, or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within computing system’s registers and/or memories into other data similarly represented as physical quantities within computing system’s memories, registers or other such information storage, transmission or display devices.

[0165] In a similar manner, term “processor” may refer to any device or portion of a device that processes electronic data from registers and/or memory and transform that electronic data into other electronic data that may be stored in registers and/or memory. As non-limiting examples, “processor” may be a CPU or a GPU. A “computing platform” may comprise one or more processors. As used herein, “software” processes may include, for example, software and/or hardware entities that perform work over time, such as tasks, threads, and intelligent agents. Also, each process may refer to multiple processes, for carrying out instructions in sequence or in parallel, continuously or intermittently. Terms “system” and “method” are used herein interchangeably insofar as system may embody one or more methods and methods may be considered a system.

[0166] In present document, references may be made to obtaining, acquiring, receiving, or inputting analog or digital data into a subsystem, computer system, or computer-implemented machine. Obtaining, acquiring, receiving, or inputting analog and digital data can be accomplished in a variety of ways such as by receiving data as a parameter of a function call or a call to an application programming interface. In some implementations, process of obtaining, acquiring, receiving, or inputting analog or digital data can be accomplished by transferring data

via a serial or parallel interface. In another implementation, process of obtaining, acquiring, receiving, or inputting analog or digital data can be accomplished by transferring data via a computer network from providing entity to acquiring entity. References may also be made to providing, outputting, transmitting, sending, or presenting analog or digital data. In various examples, process of providing, outputting, transmitting, sending, or presenting analog or digital data can be accomplished by transferring data as an input or output parameter of a function call, a parameter of an application programming interface or interprocess communication mechanism.

[0167] Although discussion above sets forth example implementations of described techniques, other architectures may be used to implement described functionality, and are intended to be within scope of this disclosure. Furthermore, although specific distributions of responsibilities are defined above for purposes of discussion, various functions and responsibilities might be distributed and divided in different ways, depending on circumstances.

[0168] Furthermore, although subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that subject matter claimed in appended claims is not necessarily limited to specific features or acts described. Rather, specific features and acts are disclosed as exemplary forms of implementing the claims.

CLAIMS

WHAT IS CLAIMED IS:

- 1 1. A computer-implemented method, comprising:
2 sending, to a client device, first video frame data for a first video frame in a
3 sequence of video frames to be displayed;
4 sending, to the client device, second video frame data for a second video frame in
5 the sequence, the second video frame data including changes in pixel data with respect to the
6 first video frame, wherein the client device is enabled to generate the second video frame for
7 display using the first video frame data and the second video frame data;
8 determining that an acknowledgement has been received indicating the client
9 device successfully received the second video frame data; and
10 sending, to the client device, third video frame data for a third video frame in the
11 sequence, wherein the third video frame data includes changes in the pixel data with respect to
12 the first video frame or the second video frame.
- 1 2. The computer-implemented method of claim 1, further comprising:
2 determining whether to include, in the third video frame data, changes with
3 respect to the first video frame or the second video frame based at least in part upon one or more
4 optimization parameters.
- 1 3. The computer-implemented method of claim 2, wherein the optimization
2 parameters relate to at least one of a data size for the changes, a number of changes, a recency of
3 transmission, or a network performance state.
- 1 4. The computer-implemented method of claim 1, further comprising:
2 storing the first video frame data and the second video frame data in a client
3 reference buffer, wherein both the first video frame data and the second video frame data are
4 available to be used by the client device as a reference frame for determining the third video
5 frame data.

1 5. The computer-implemented method of claim 1, further comprising:
2 receiving, from the client device, an acknowledgement indicating that the client
3 device successfully received the third video frame data; and
4 causing the third video frame to be available, along with the first video frame and
5 the second video frame, for use as a reference frame for one or more subsequent video frames in
6 the sequence.

1 6. The computer-implemented method of claim 1, further comprising:
2 receiving, from the client device, a request to remove the first video frame from a
3 client reference buffer on the client device; and
4 sending, to the client device, an acknowledgement that the first video frame is to
5 be removed from consideration as a reference frame for one or more subsequent video frames in
6 the sequence and can be removed from the client reference buffer.

1 7. The computer-implemented method of claim 1, further comprising:
2 determining to remove at least one of the first video frame data, the second video
3 frame data, and the third video frame data from a client reference buffer on the client device
4 based at least in part upon a recency or a diversity of data in the client reference buffer.

1 8. The computer-implemented method of claim 1, further comprising:
2 encoding the first video frame data, the second video frame data, and the third
3 video frame data before sending to the client device.

1 9. A video streaming system, comprising:
2 a streaming server to transmit a stream of video data to a client device; and
3 an encoder to encode a sequence of video frame data to be included in the stream,
4 wherein the encoder is to encode the sequence at least in part by:
5 selecting a reference frame to use for a current frame in the video stream,
6 the reference frame corresponding to a confirmed frame, of one or more confirmed

7 frames, stored to a reference buffer for which video data is acknowledged to have been
8 received by the client device; and
9 encoding video data for the current frame including changes in pixel data
10 relative to the selected reference frame.

1 10. The video streaming system of claim 9, wherein the encoder is further to
2 select the reference frame using at least one selection criterion.

1 11. The video streaming system of claim 9, wherein the at least one selection
2 criterion includes at least one of a data size, a number of changes, a recency of transmission, or a
3 network performance state.

1 12. The video streaming system of claim 9, wherein the encoder is further to
2 receive, from the client device, acknowledgement that the current frame was properly received and
3 decoded by the client device, and wherein data for the current frame is to be stored to the reference
4 buffer as an additional confirmed frame.

1 13. The video streaming system of claim 9, wherein the encoder is to remove,
2 from the reference buffer, one or more confirmed frames indicated to no longer be available for
3 use as a reference frame.

1 14. The video streaming system of claim 9, wherein the encoder is to apply
2 forward error correction to the video data before the video data is transmitted to the client device.

1 15. The video streaming system of claim 10, wherein the system comprises at
2 least one of:
3 a system for performing simulation operations;
4 a system for performing simulation operations to test or validate autonomous
5 machine applications;

6 a system for rendering graphical output;
7 a system for performing deep learning operations;
8 a system implemented using an edge device;
9 a system incorporating one or more Virtual Machines (VMs);
10 a system implemented at least partially in a data center; or
11 a system implemented at least partially using cloud computing resources.

1 16. A non-transitory computer-readable storage medium including
2 instructions that, if executed by one or more processor, cause the one or more processors to:
3 select a reference frame to use for a current frame in the video stream, the
4 reference frame corresponding to a confirmed frame, of one or more confirmed frames, stored to
5 a reference buffer for which video data is acknowledged to have been received by the client
6 device; and
7 encode video data for the current frame including changes in pixel data relative to
8 the selected reference frame.

1 17. The non-transitory computer-readable storage medium of claim 16,
2 wherein the encoder is further to select the reference frame using at least one selection criterion.

1 18. The non-transitory computer-readable storage medium of claim 16, wherein
2 the at least one selection criterion includes at least one of a data size, a number of changes, a
3 recency of transmission, or a network performance state.

1 19. The non-transitory computer-readable storage medium of claim 16, wherein
2 the encoder is further to receive, from the client device, acknowledgement that the current frame
3 has been properly received and decoded by the client device, and wherein data for the current
4 frame is to be stored to the reference buffer as an additional confirmed frame.

1 20. The non-transitory computer-readable storage medium of claim 16,
2 wherein the encoder is to remove, from the reference buffer, one or more confirmed frames
3 indicated to no longer be available for use as a reference frame.

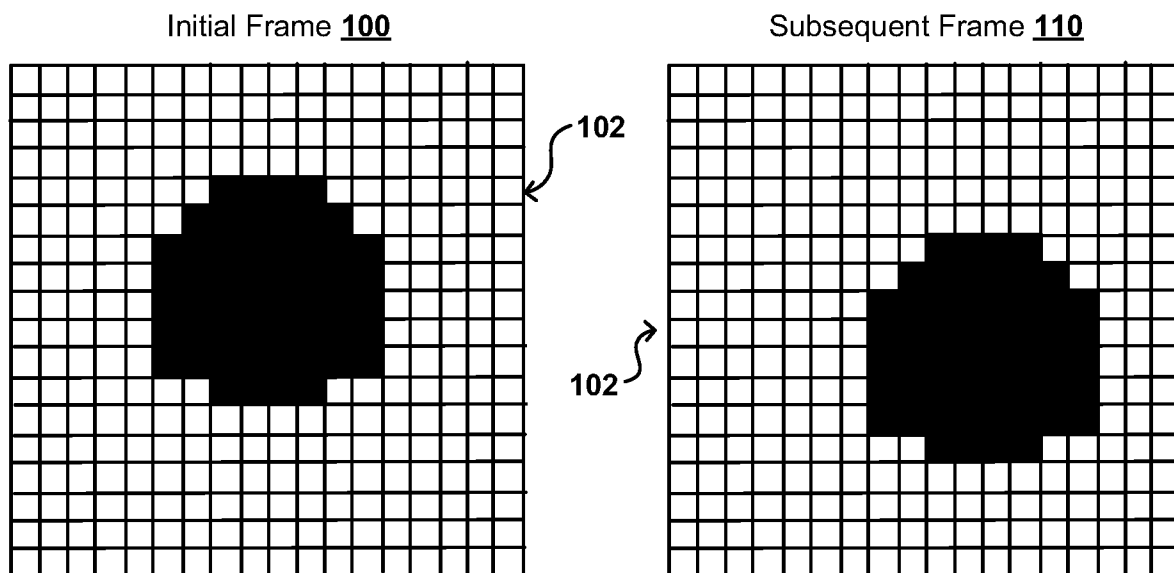


FIG. 1A

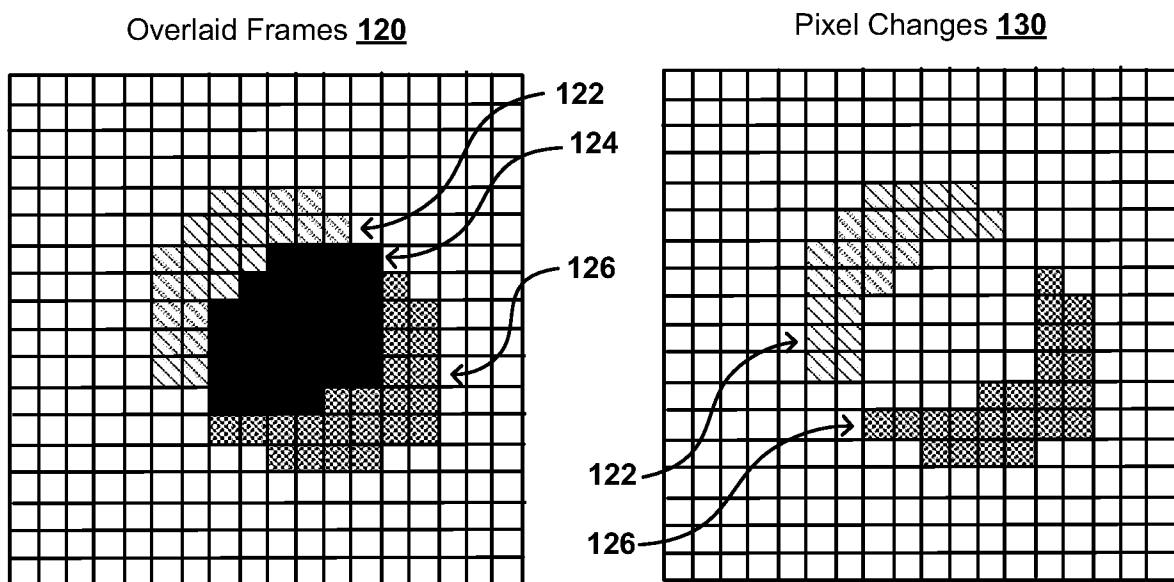


FIG. 1B

FIG. 1C

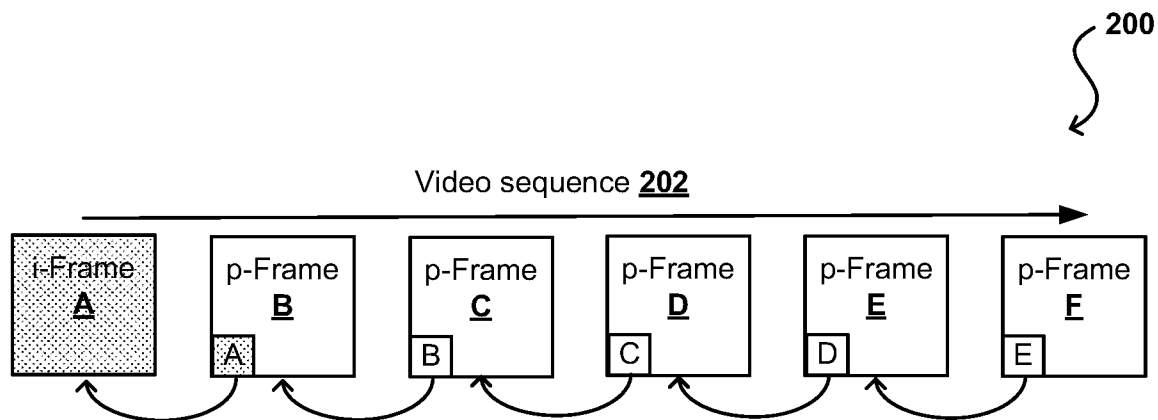


FIG. 2A

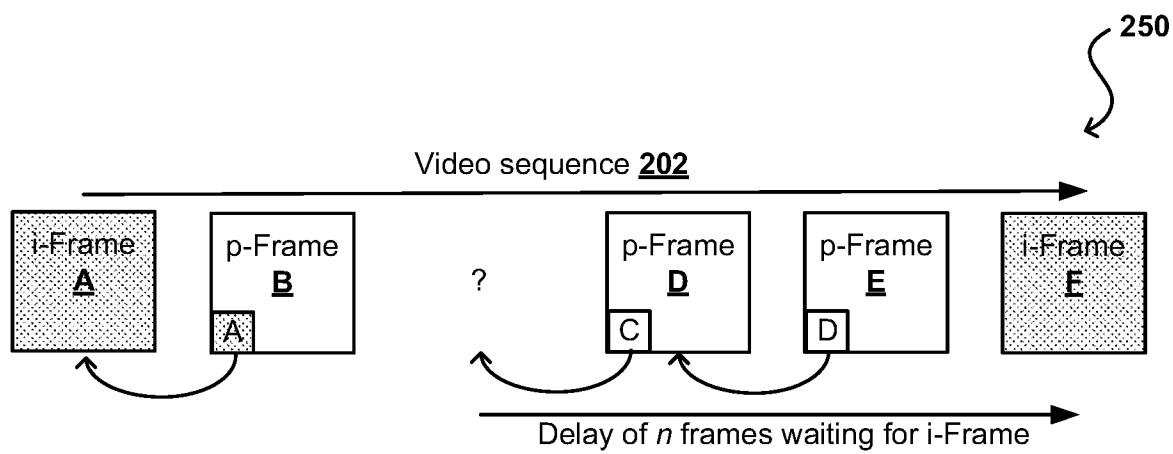


FIG. 2B

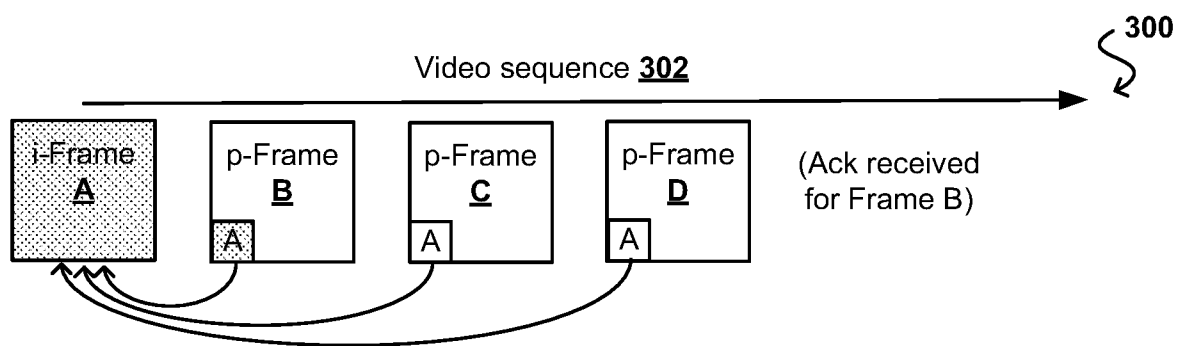


FIG. 3A

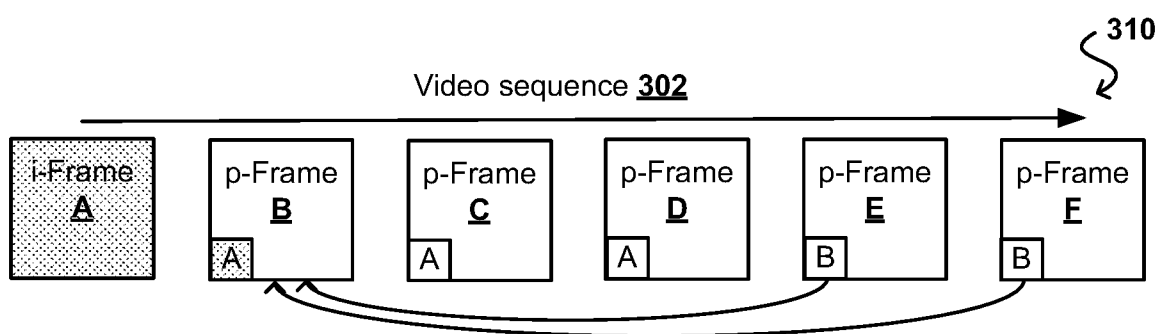


FIG. 3B

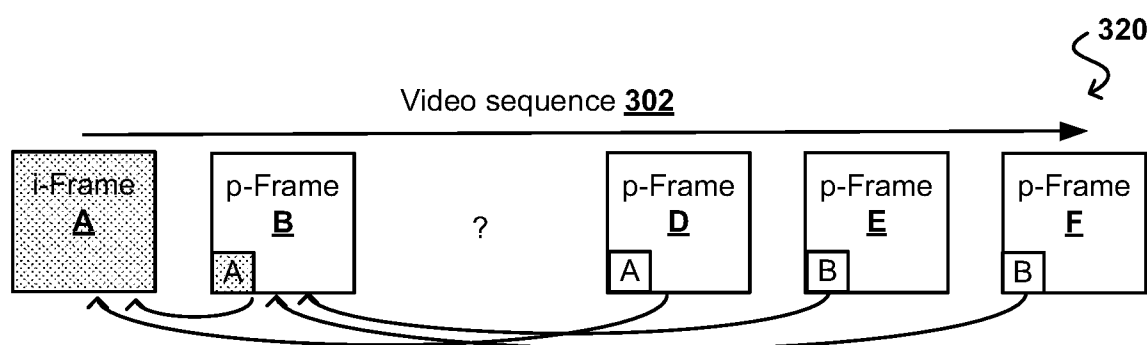


FIG. 3C

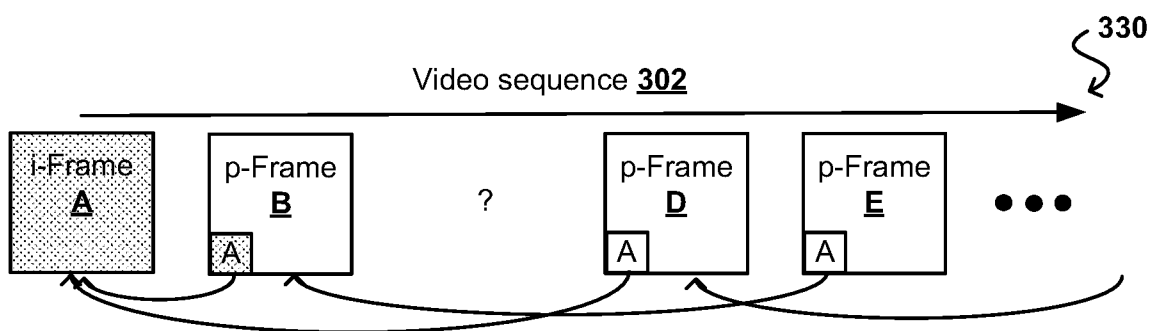


FIG. 3D

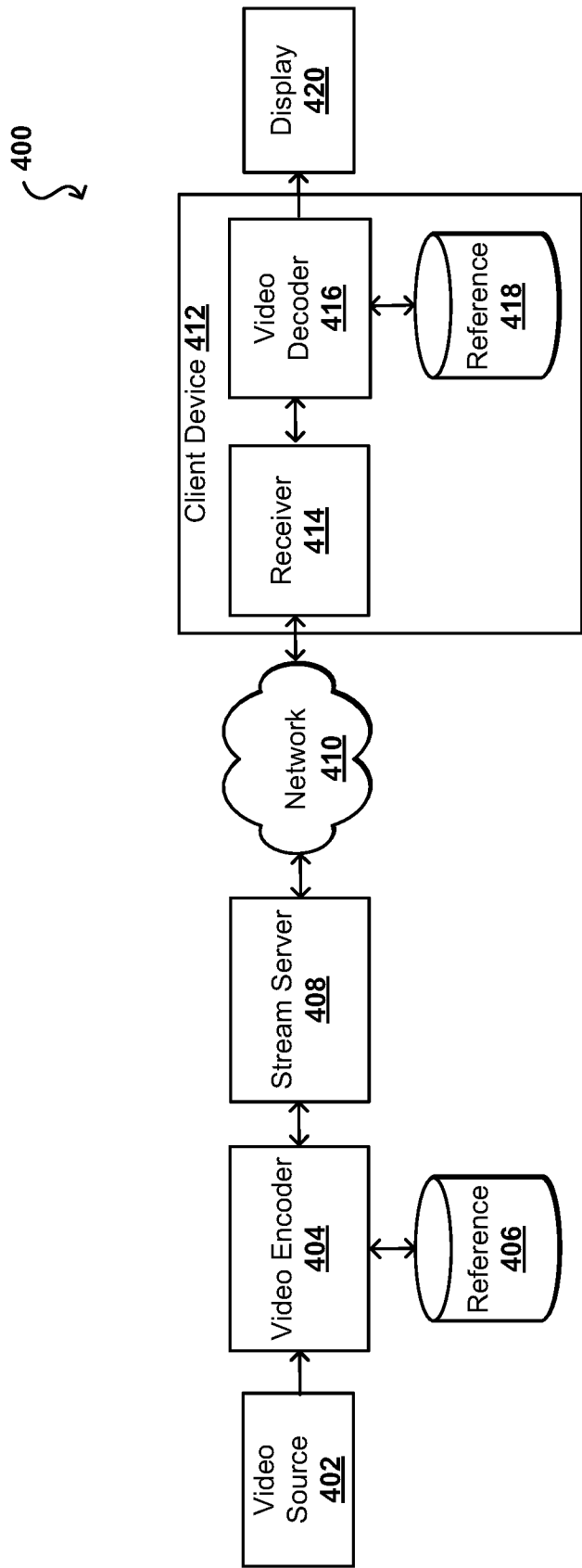


FIG. 4

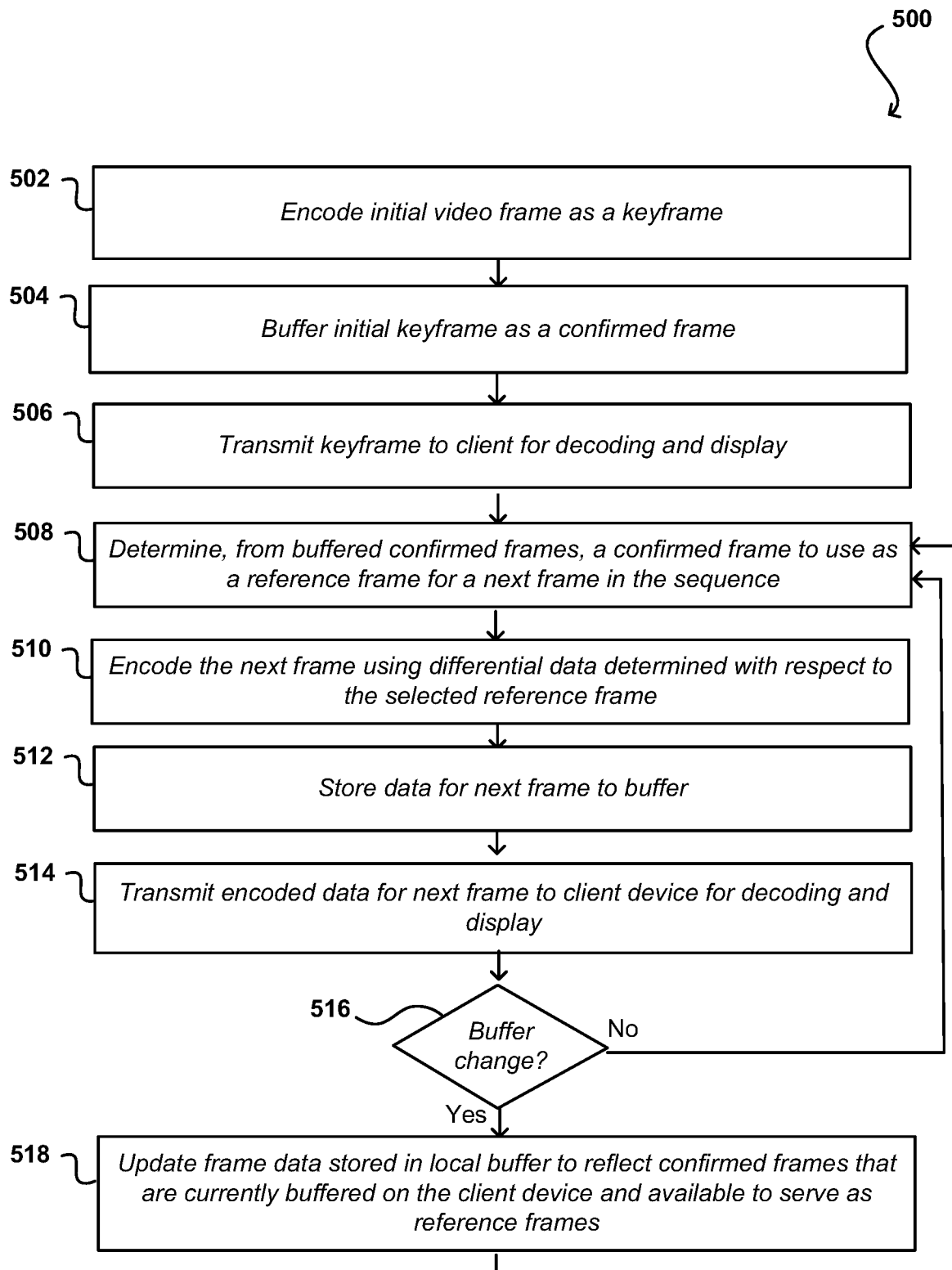


FIG. 5A

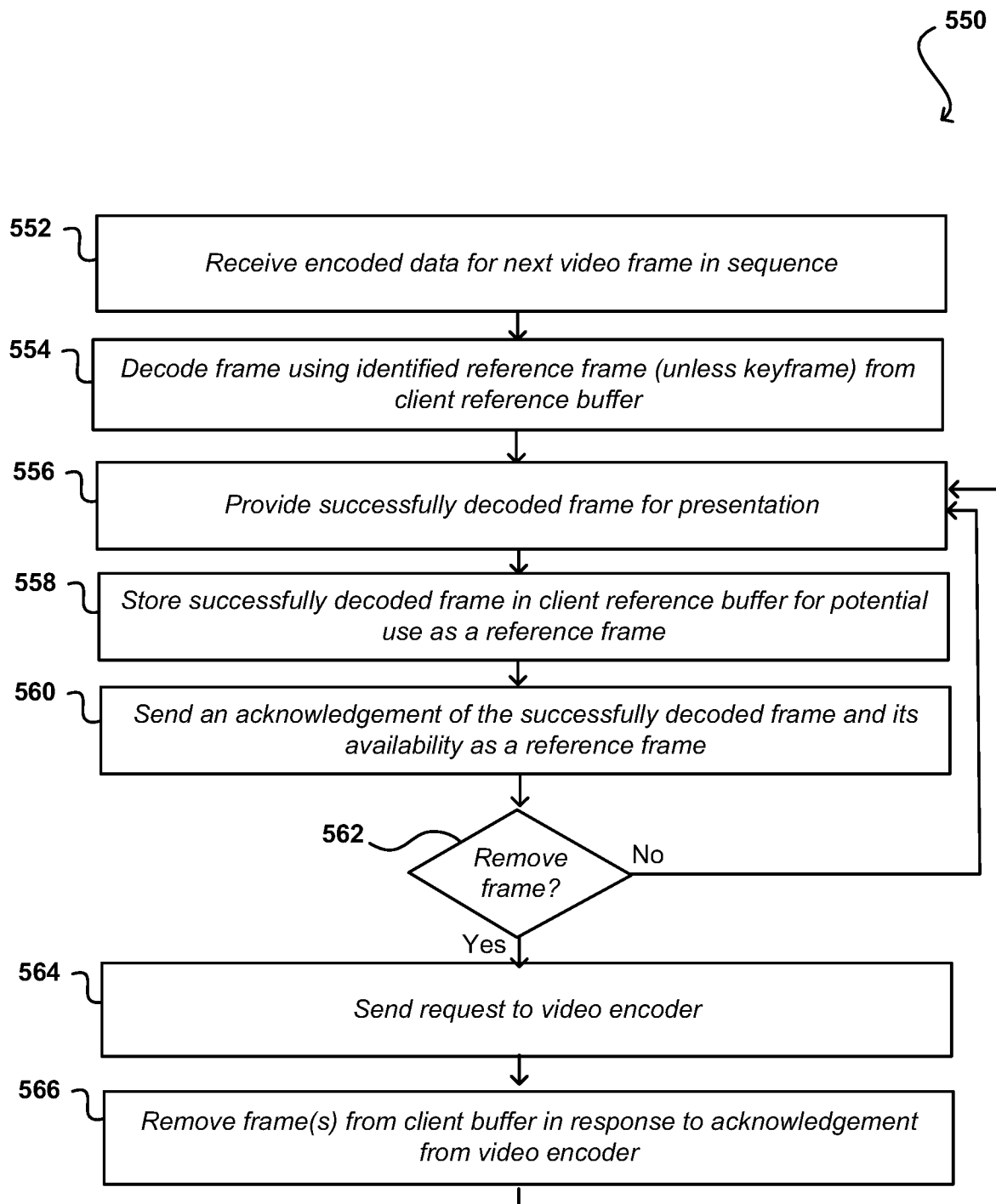


FIG. 5B

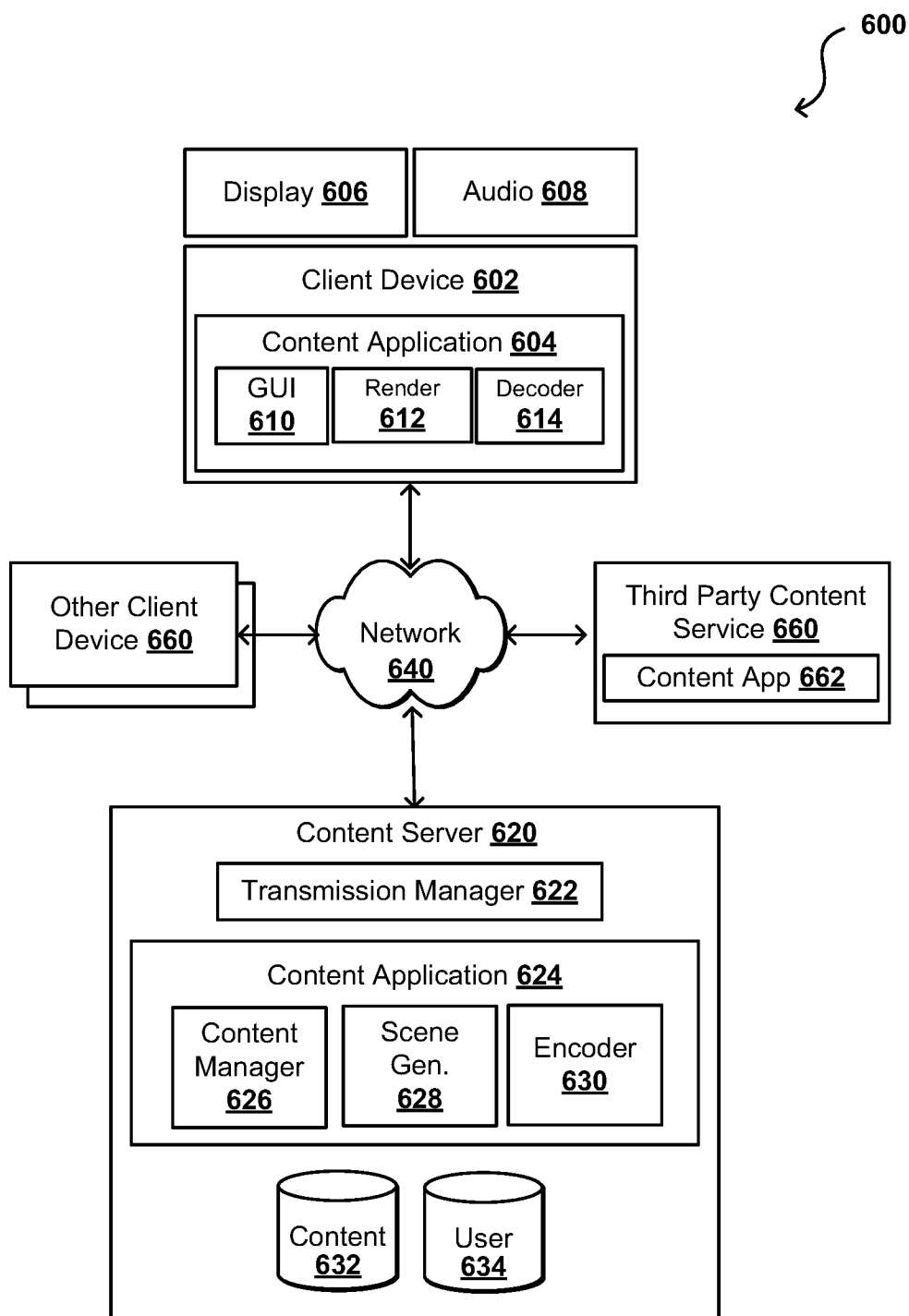


FIG. 6

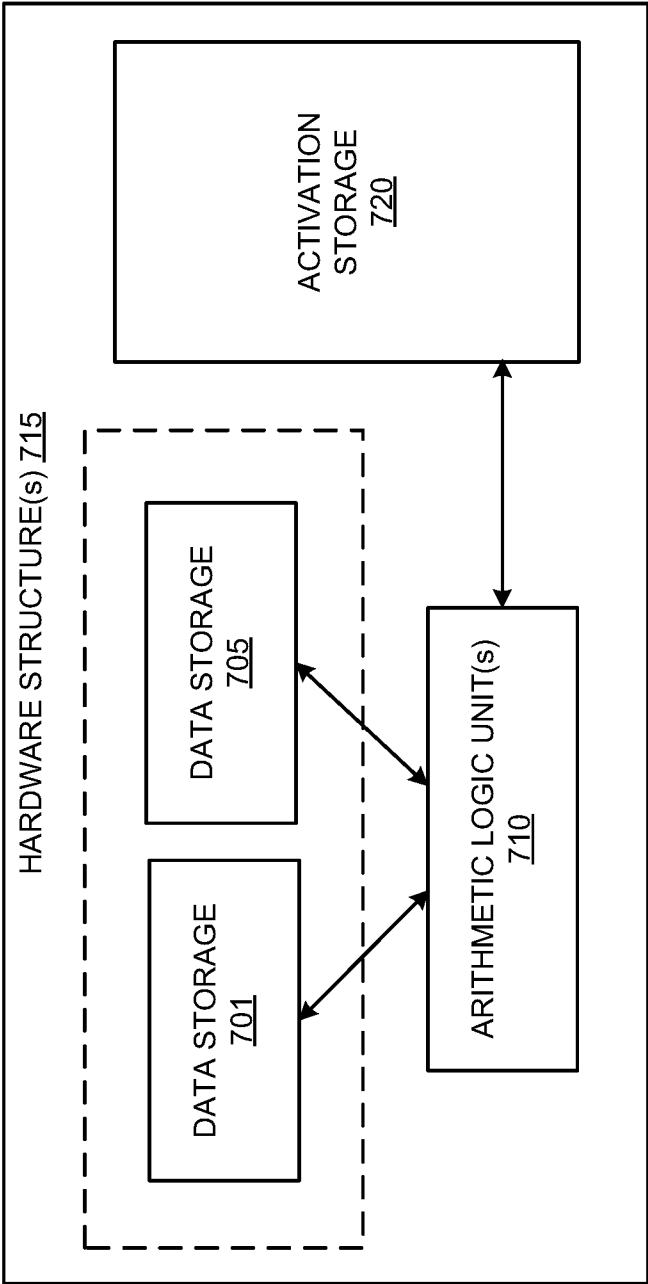


FIG. 7A

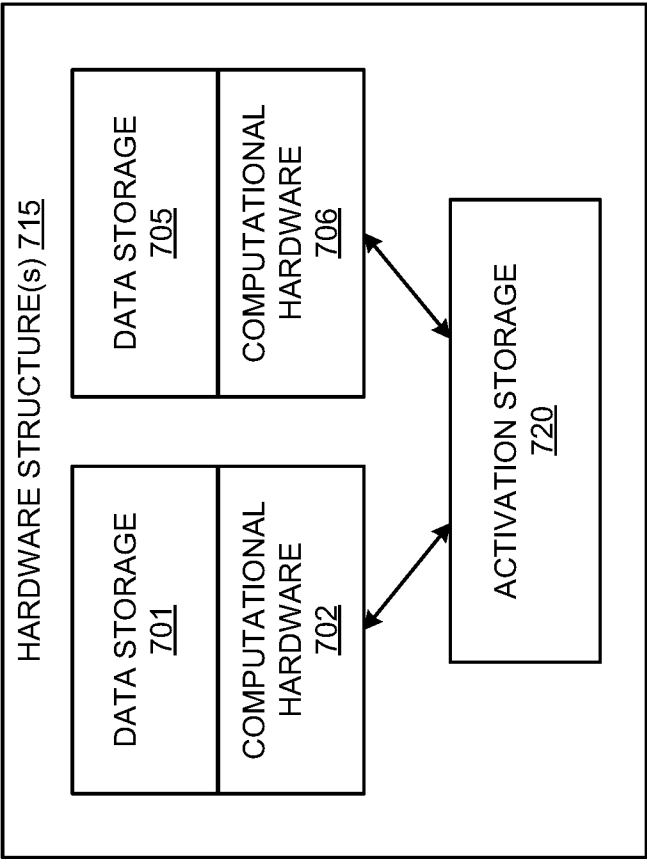


FIG. 7B

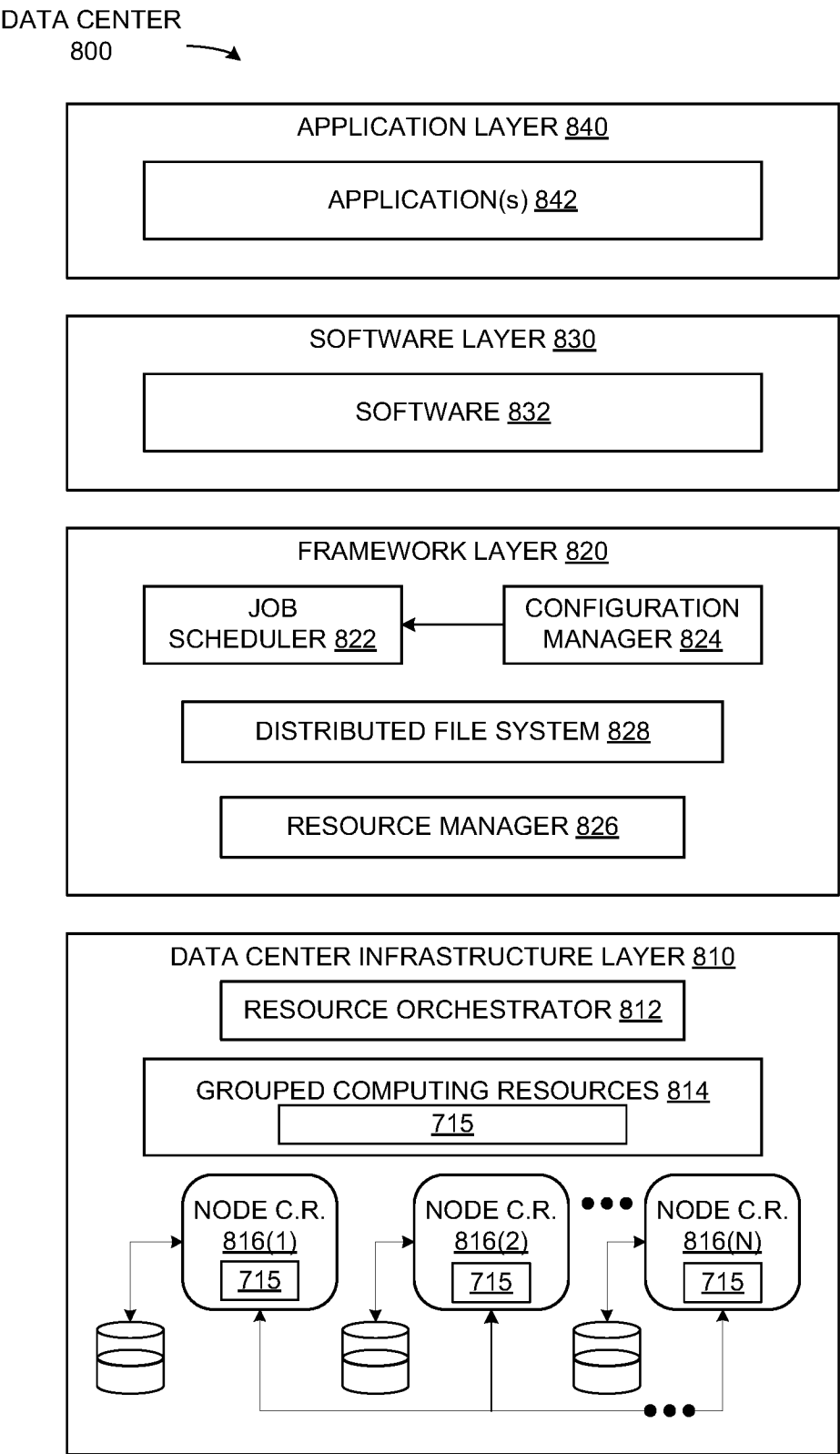


FIG. 8

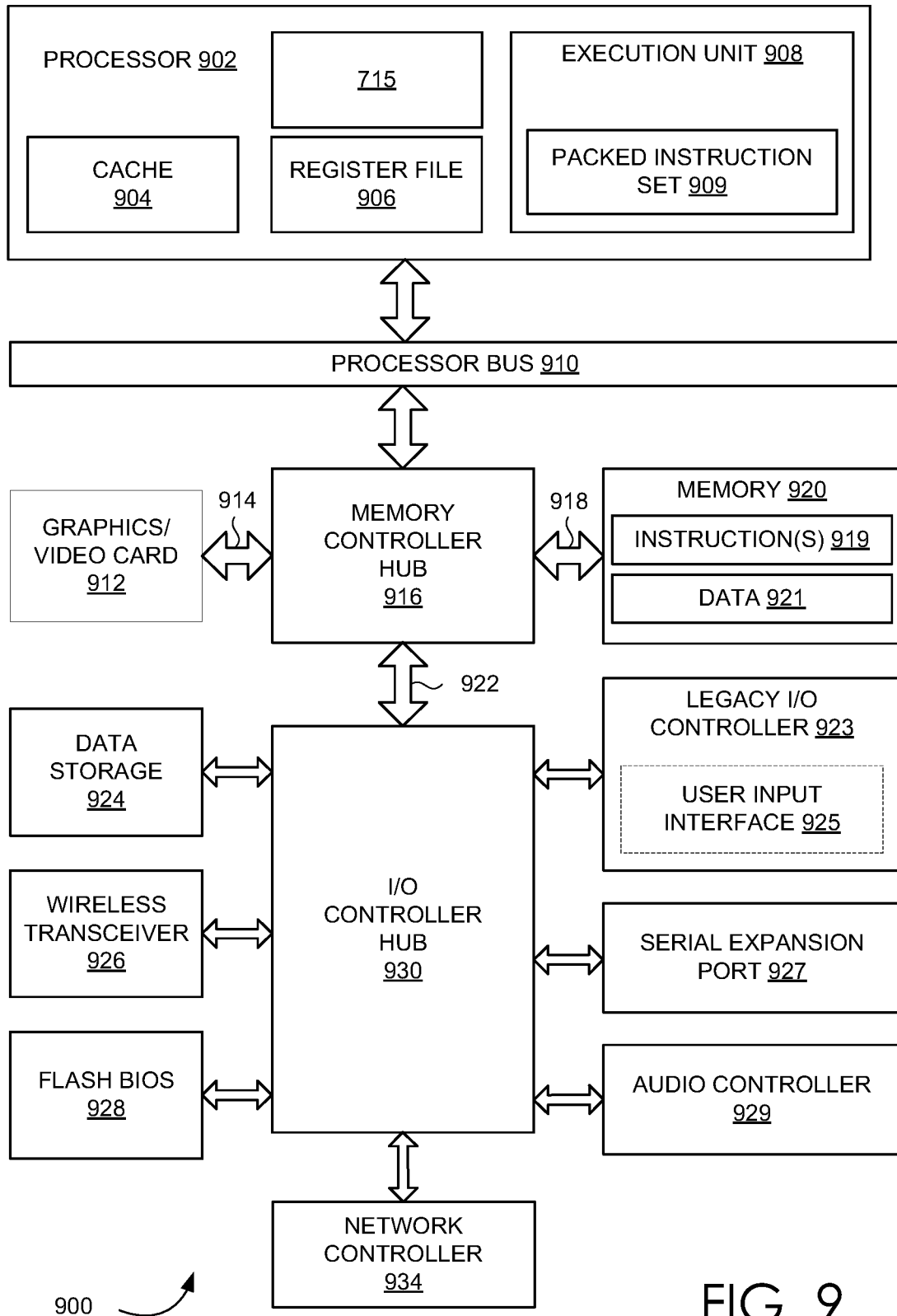


FIG. 9

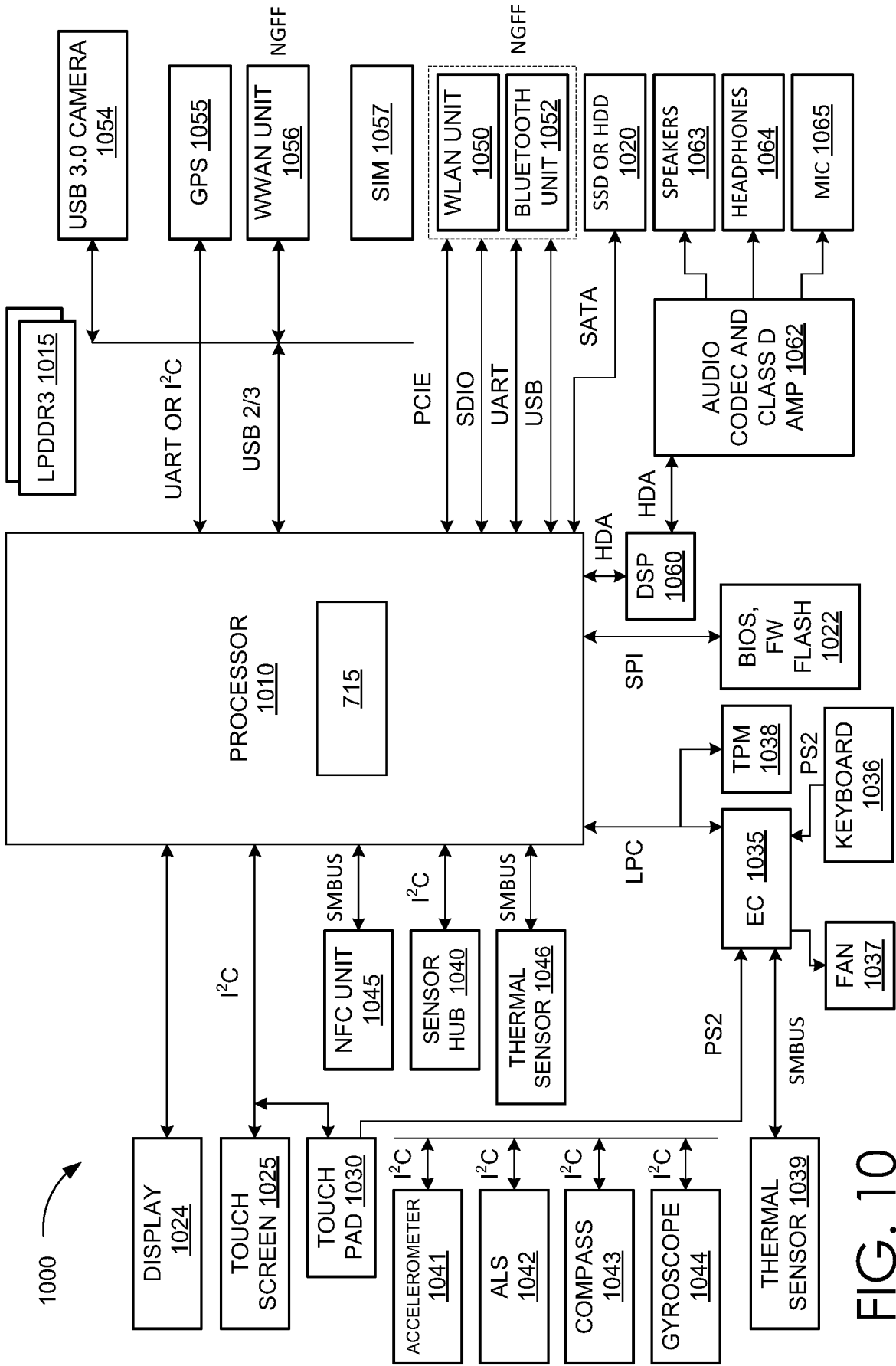


FIG. 10

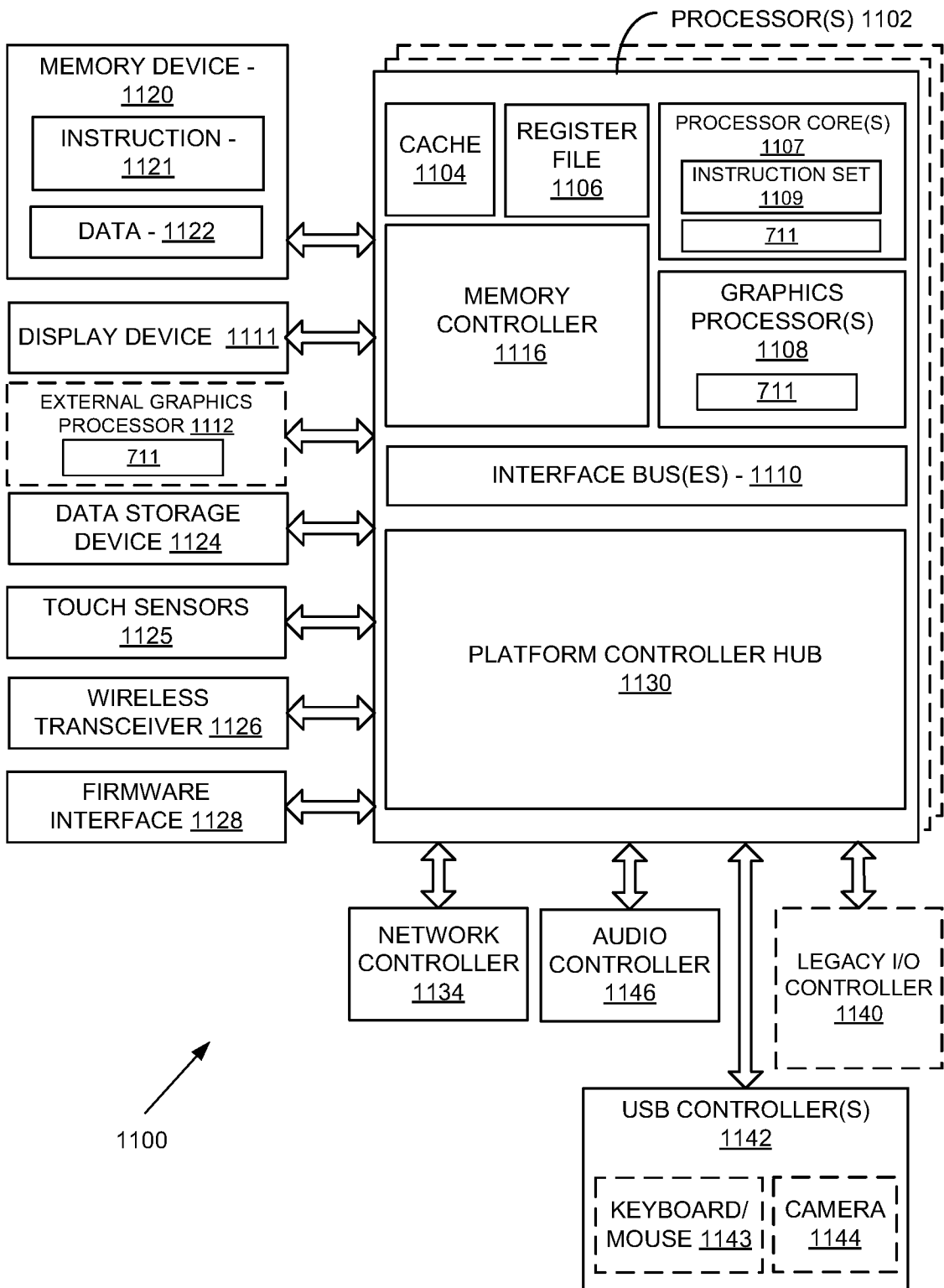


FIG. 11

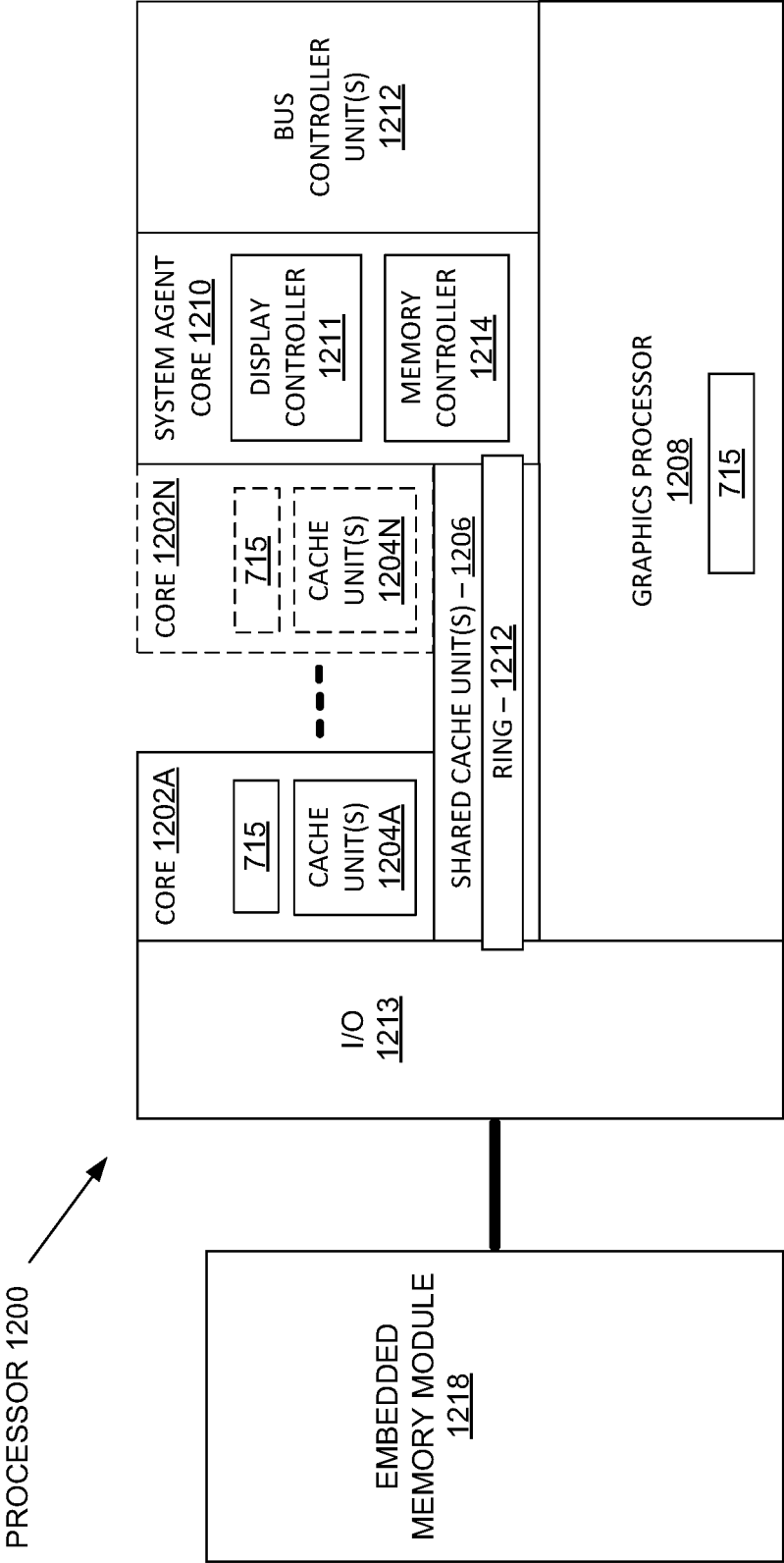


FIG. 12

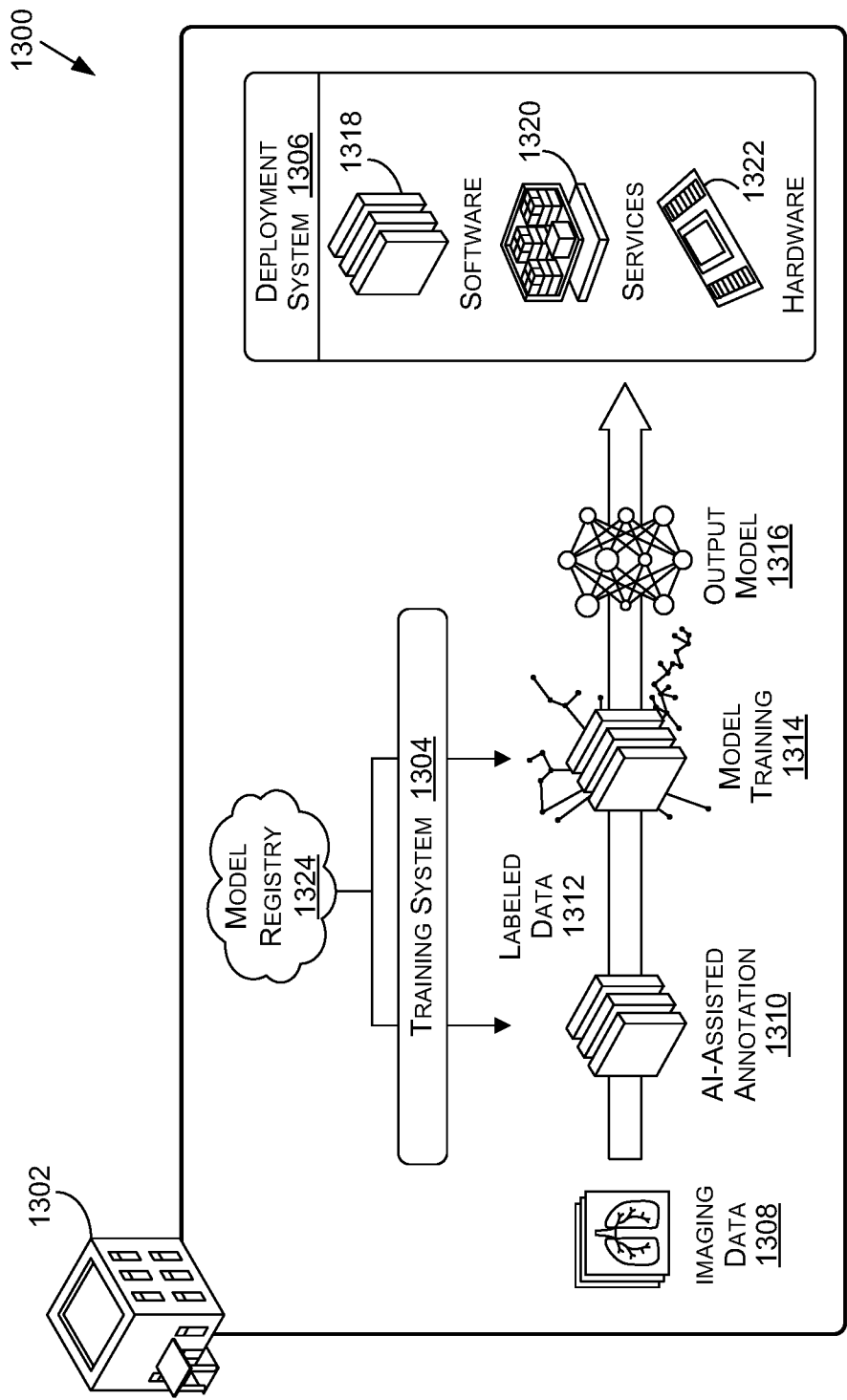


FIG. 13

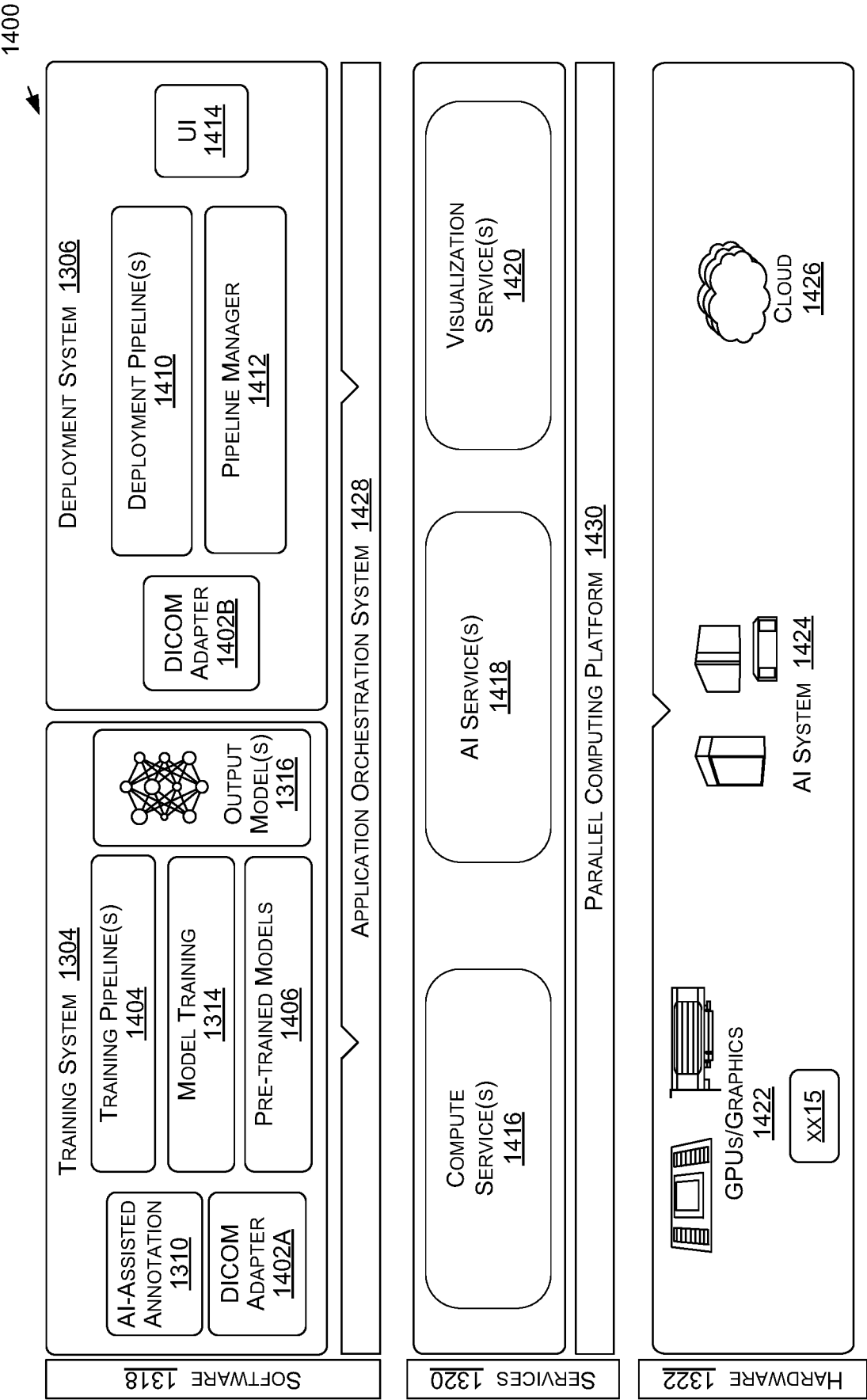


FIG. 14

1500

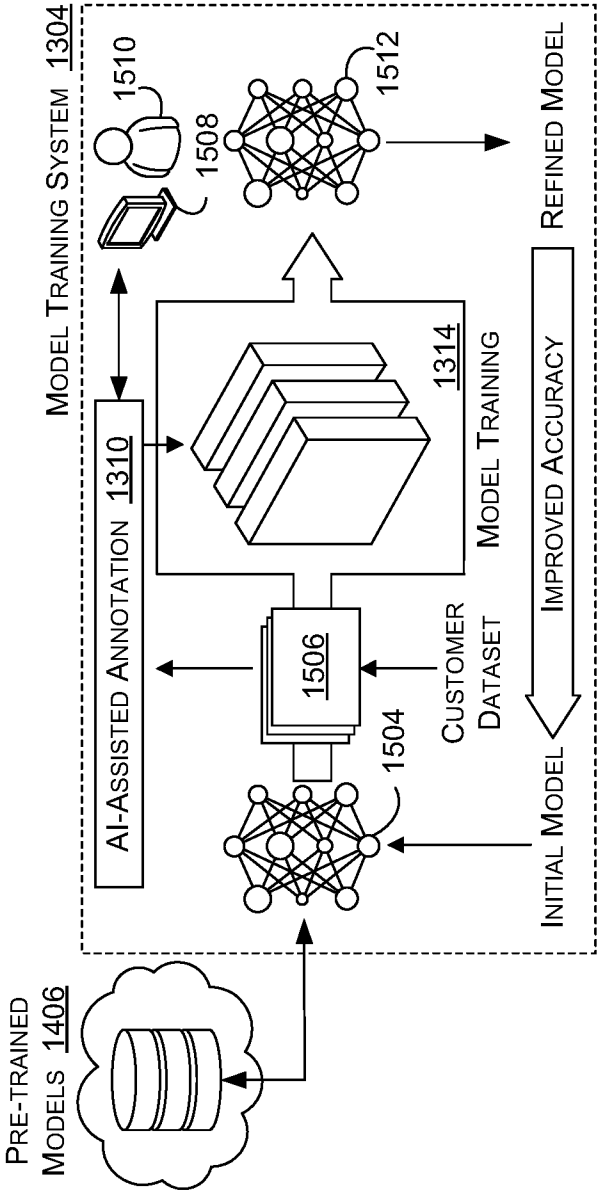


FIG. 15A

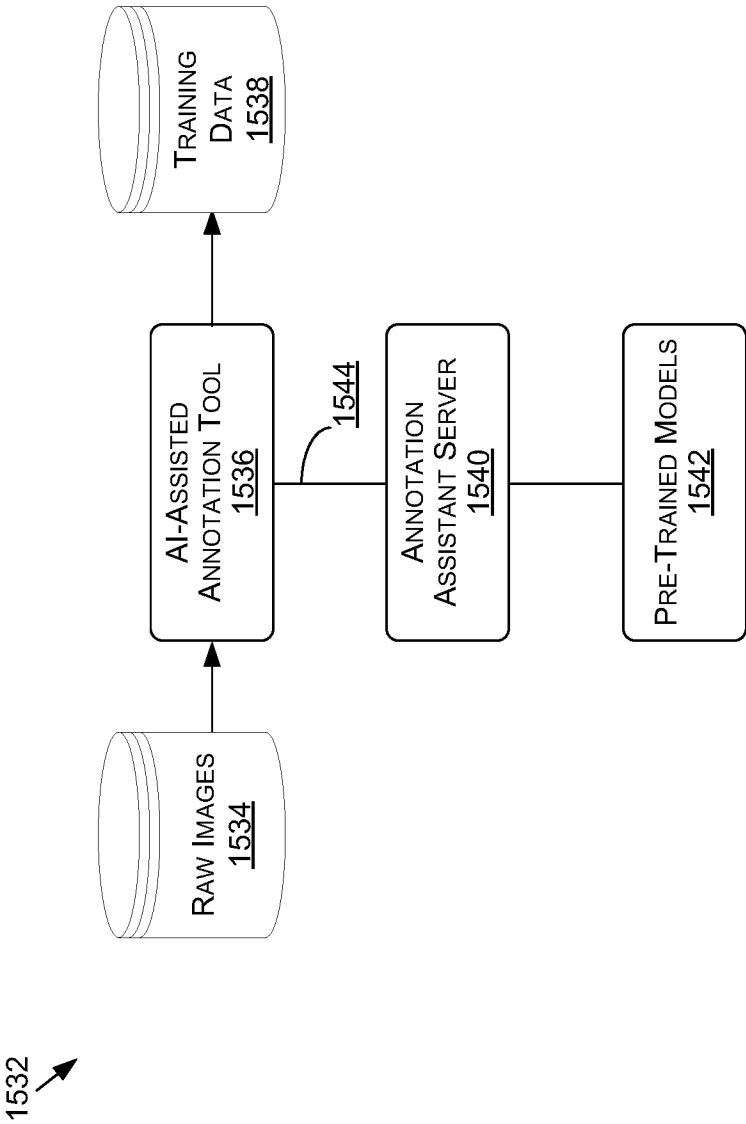


FIG. 15B

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/036476

A. CLASSIFICATION OF SUBJECT MATTER

INV. **H04N19/164 H04N19/105 H04N19/172 H04N19/61 H04N19/89**
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2021/120232 A1 (SRIKANTAM SRI RANJAN [US] ET AL) 22 April 2021 (2021-04-22) paragraph [0022] - paragraph [0027]; figure 1 paragraph [0029] paragraph [0031] - paragraph [0041]; figure 1 paragraph [0042] - paragraph [0044] paragraph [0056] - paragraph [0060] paragraph [0072] - paragraph [0075] -----	1-20
X	US 2011/026414 A1 (BANERJEE DEBARAG [US]) 3 February 2011 (2011-02-03) paragraph [0022] - paragraph [0025]; figure 1 paragraph [0039] - paragraph [0043] ----- -/--	1-20



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

23 September 2022

Date of mailing of the international search report

05/10/2022

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Moschetti, Fulvio

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2022/036476

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014/286407 A1 (MONTMINY CHRISTIAN JOSEPH [CA] ET AL) 25 September 2014 (2014-09-25) paragraph [0013] - paragraph [0018]; figure 1 paragraph [0023] - paragraph [0030]; figure 2 -----	1-20
X	US 2012/219067 A1 (JEFREMOV ANDREI [SE] ET AL) 30 August 2012 (2012-08-30) paragraph [0014] - paragraph [0017] paragraph [0031] - paragraph [0036]; figures 3-4 paragraph [0041] - paragraph [0042] paragraph [0047] - paragraph [0055]; figures 6-7 -----	1-20

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2022/036476

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2021120232 A1	22-04-2021	US 2021120232 A1 WO 2022139902 A1	22-04-2021 30-06-2022
US 2011026414 A1	03-02-2011	US 2011026414 A1 US 2011228845 A1 WO 2011016988 A2	03-02-2011 22-09-2011 10-02-2011
US 2014286407 A1	25-09-2014	NONE	
US 2012219067 A1	30-08-2012	CN 103430538 A EP 2681915 A1 US 2012219067 A1 WO 2012113763 A1	04-12-2013 08-01-2014 30-08-2012 30-08-2012