US012383838B2

(12) **United States Patent**
Yerva et al.

(10) **Patent No.: US 12,383,838 B2**
(45) **Date of Patent: *Aug. 12, 2025**

(54) **GAME EVENT RECOGNITION FOR USER GENERATED CONTENT**

(71) Applicant: **Nvidia Corporation**, Santa Clara, CA (US)

(72) Inventors: **Suresh Yerva**, Beglauru (IN); **Stephen Holmes**, Fort Collins, CO (US)

(73) Assignee: **Nvidia Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **18/317,639**

(22) Filed: **May 15, 2023**

(65) **Prior Publication Data**

US 2023/0277944 A1 Sep. 7, 2023

Related U.S. Application Data

(63) Continuation of application No. 17/075,377, filed on Oct. 20, 2020, now Pat. No. 11,648,481.

(51) **Int. Cl.**
A63F 13/77 (2014.01)
A63F 13/537 (2014.01)
(Continued)

(52) **U.S. Cl.**
CPC ***A63F 13/77*** (2014.09); ***A63F 13/537*** (2014.09); ***H04N 21/4394*** (2013.01); ***H04N 21/4781*** (2013.01)

(58) **Field of Classification Search**
CPC ... A63F 13/77; A63F 13/537; H04N 21/4394; H04N 21/4781
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,445,549 B1 * 11/2008 Best A63F 13/12 463/31
8,556,729 B2 * 10/2013 Suzuki A63F 13/803 463/43
(Continued)

FOREIGN PATENT DOCUMENTS

CN 105409224 A 3/2016
CN 106126097 A 11/2016
(Continued)

OTHER PUBLICATIONS

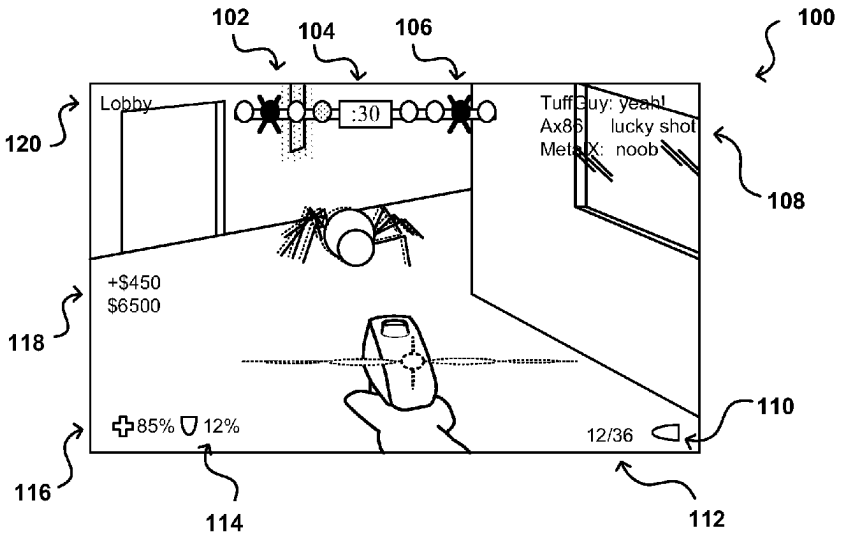
Non-Final Office Action dated Aug. 12, 2021 issued in U.S. Appl. No. 17/075,377.
(Continued)

Primary Examiner — Malina D. Blaise
(74) *Attorney, Agent, or Firm* — Hogan Lovells US LLP

(57) **ABSTRACT**

Automated detection of events in content can be performed using regions of information associated with various user interface or display elements. Certain elements can be indicative of a type of event, and regions associated with these elements can be analyzed on a per-frame basis. If one of these primary regions shows a state or transition that is indicative of one of these events, one or more secondary regions can be analyzed as well to attempt to verify whether that event occurred, as well as whether that event qualifies for selection for additional use. Selected events can be used for purposes such as to generate highlight montages, training videos, or user profiles. These events may be positioned at different layers of an event hierarchy, where child regions are only analyzed for frames where a parent region is indicative of a type of event.

20 Claims, 18 Drawing Sheets



(51) **Int. Cl.**
H04N 21/439 (2011.01)
H04N 21/478 (2011.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,632,410	B2 *	1/2014	Perlman	H04N 19/146 463/42
10,923,157	B2 *	2/2021	Hendry	G11B 27/3036
11,071,919	B2 *	7/2021	Willette	A63F 13/85
2007/0270215	A1 *	11/2007	Miyamoto	A63F 13/45 463/32
2011/0307833	A1 *	12/2011	Dale	G06F 3/04886 715/835
2013/0179308	A1 *	7/2013	Agustin	G06Q 30/0641 705/27.1
2014/0018165	A1 *	1/2014	Kern	A63F 13/358 463/31
2014/0155156	A1 *	6/2014	Peck	A63F 13/847 463/31
2016/0279509	A1 *	9/2016	Miller	A63F 13/26
2016/0345035	A1 *	11/2016	Han	H04N 21/21805
2021/0086075	A1 *	3/2021	Cockram	G06T 13/80

FOREIGN PATENT DOCUMENTS

CN	109672922	A	4/2019
CN	111265859	A	6/2020

OTHER PUBLICATIONS

Final Office Action dated Dec. 7, 2021 issued in U.S. Appl. No. 17/075,377.
Non-Final Office Action dated May 6, 2022 issued in U.S. Appl. No. 17/075,377.
Notice of Allowance dated Dec. 29, 2022 issued in U.S. Appl. No. 17/075,377.

* cited by examiner

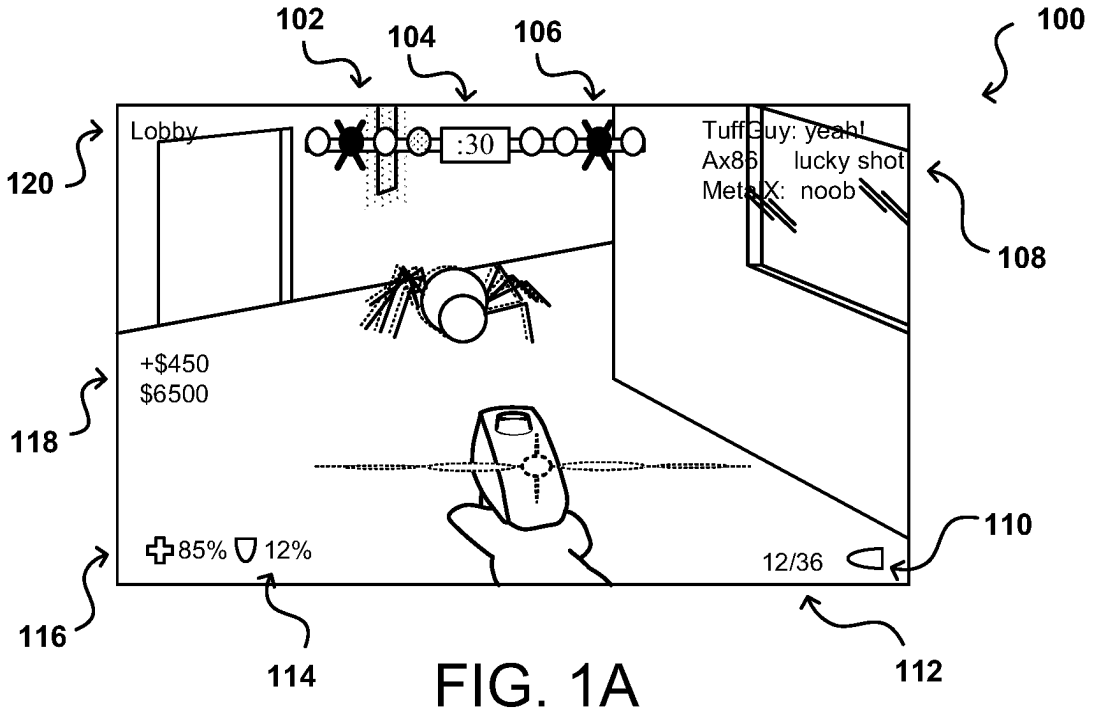

102
104
106
100
Lobby
:30
TuffGuy: yeah!
Ax86: lucky shot
MetaX: noob
120
108
+$450
$6500
118
110
85% 12%
12/36
116
114
112

FIG. 1A

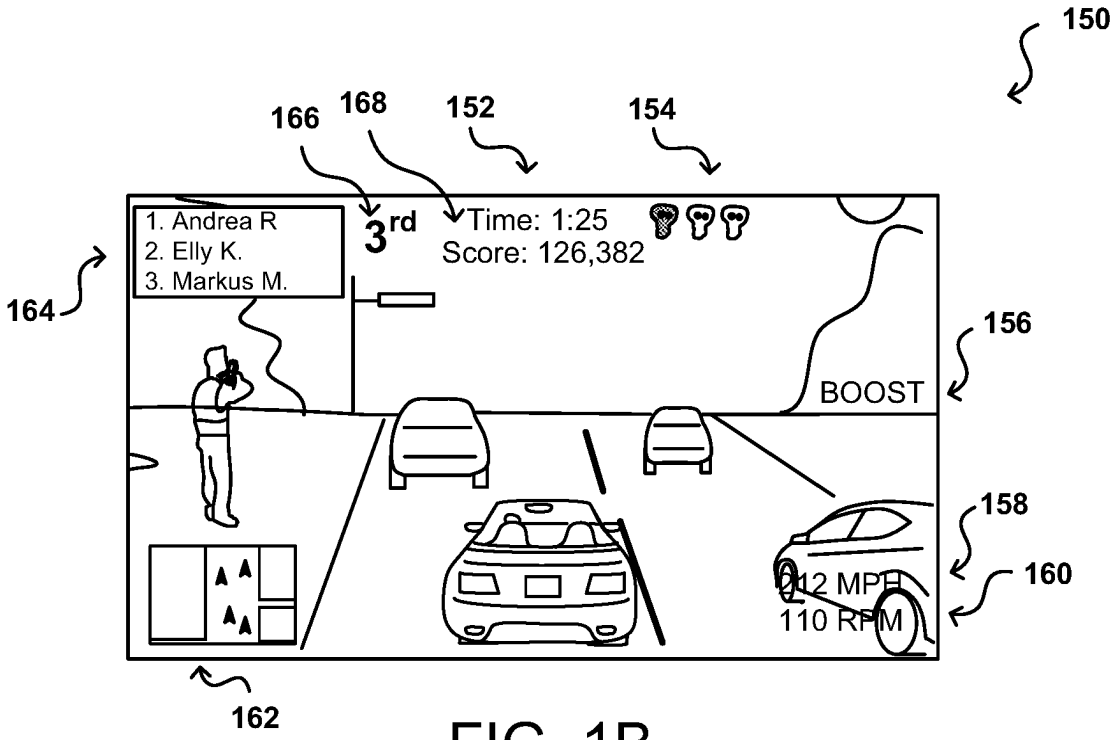

150
166
168
152
154
1. Andrea R
2. Elly K.
3. Markus M.
3rd
Time: 1:25
Score: 126,382
164
156
BOOST
158
160
112 MPH
110 RPM
162

FIG. 1B

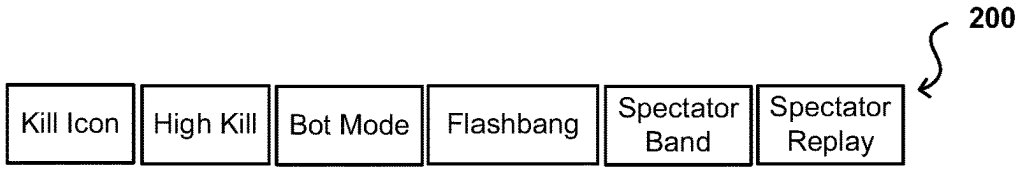


FIG. 2A

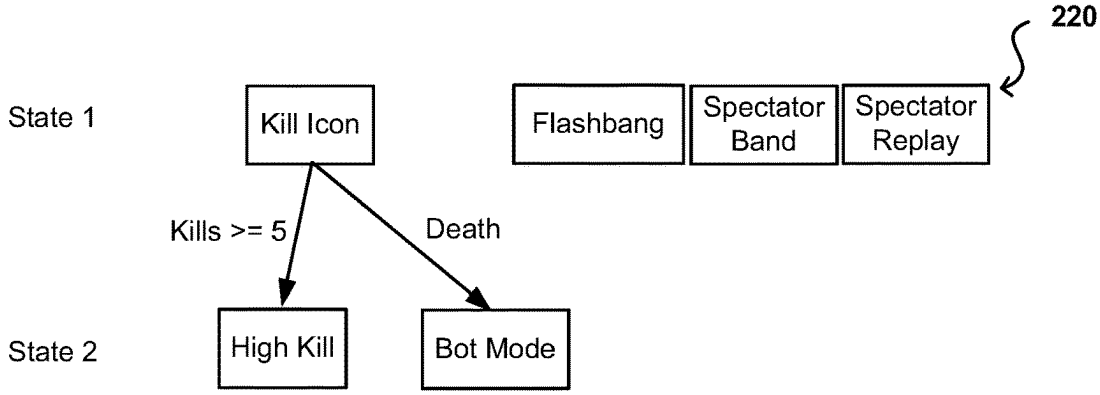


FIG. 2B

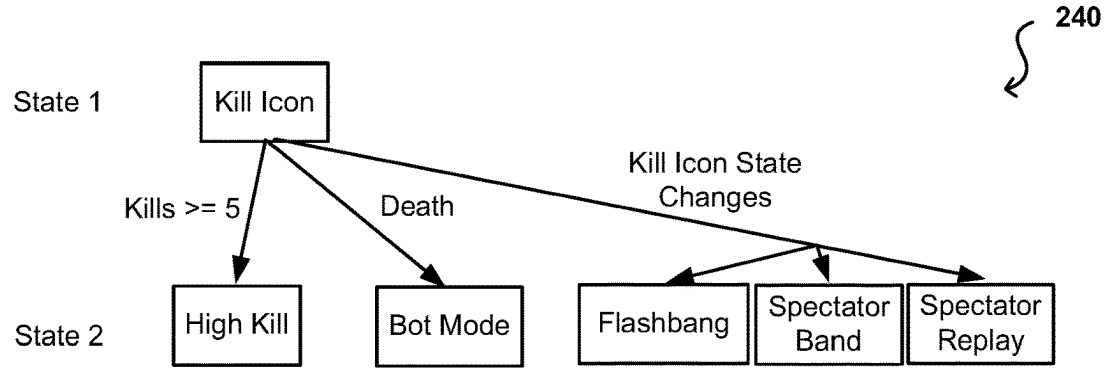


FIG. 2C

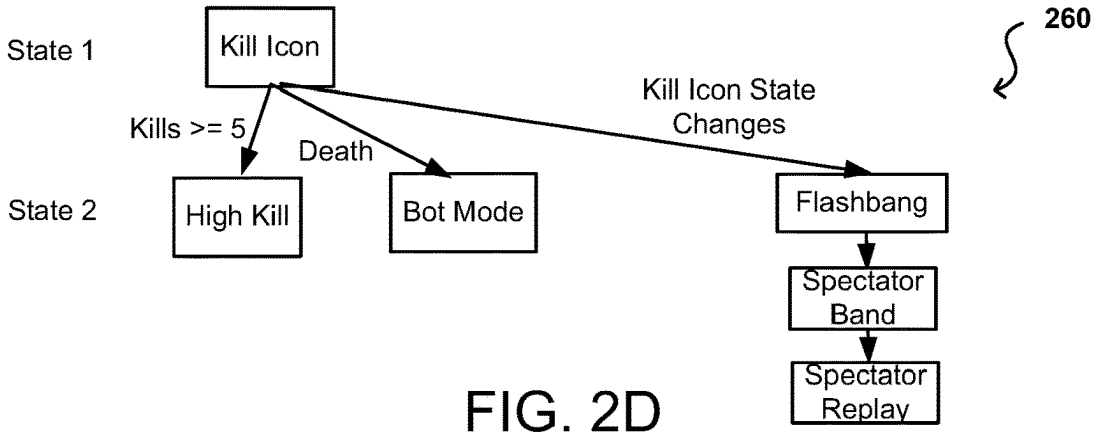


FIG. 2D

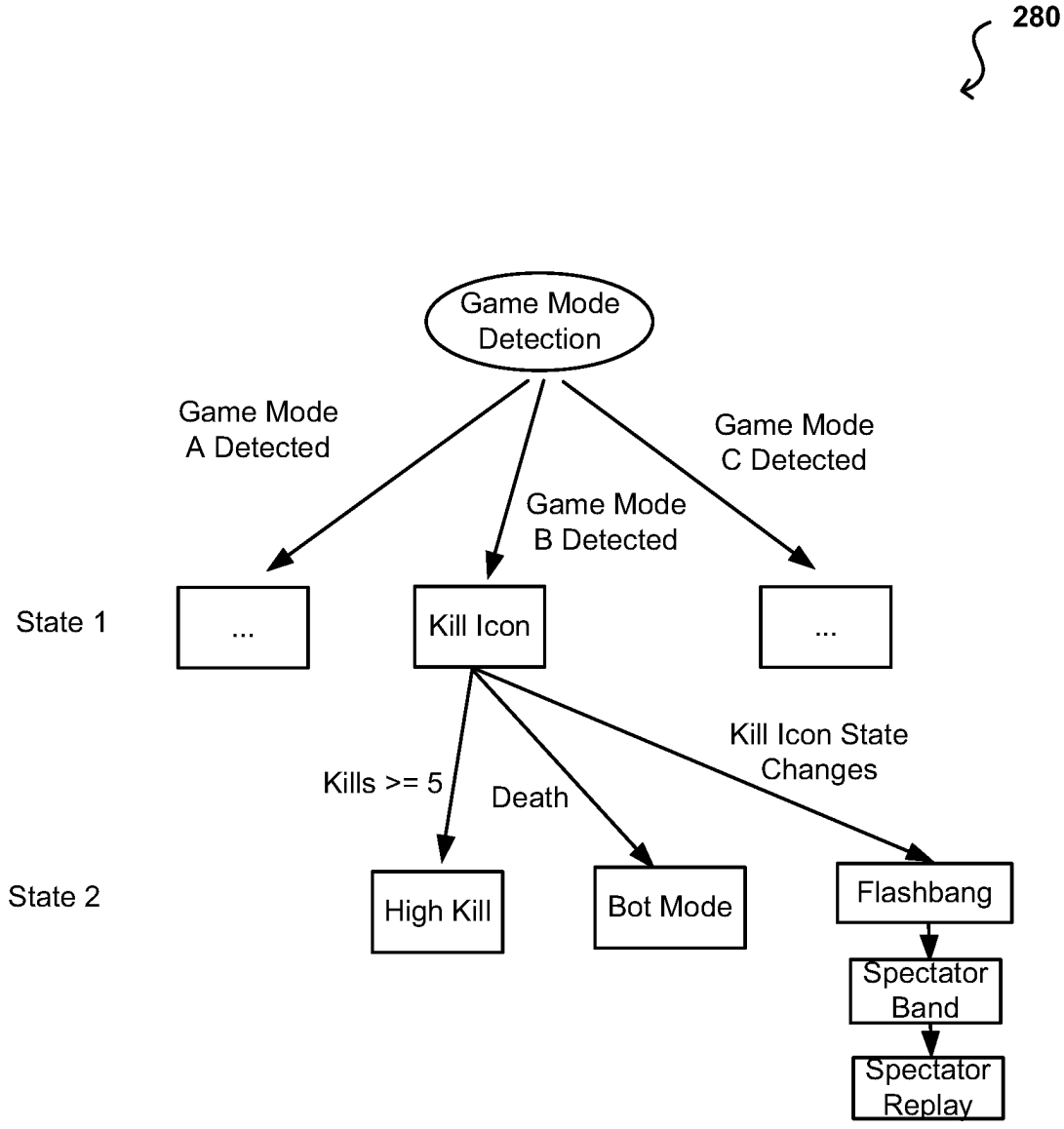

280
Game Mode Detection
Game Mode A Detected
Game Mode B Detected
Game Mode C Detected
State 1
...
Kill Icon
...
Kills >= 5
Death
Kill Icon State Changes
State 2
High Kill
Bot Mode
Flashbang
Spectator Band
Spectator Replay

FIG. 2E

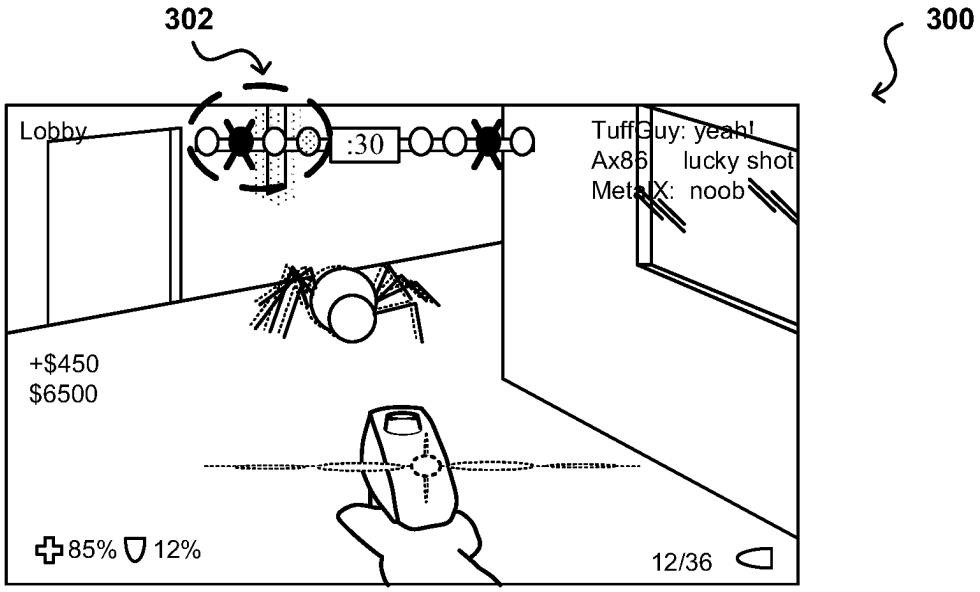

302
300
Lobby
:30
TuffGuy: yeah!
Ax86 lucky shot
MetaX: noob
+$450
$6500
85% 12%
12/36

FIG. 3A

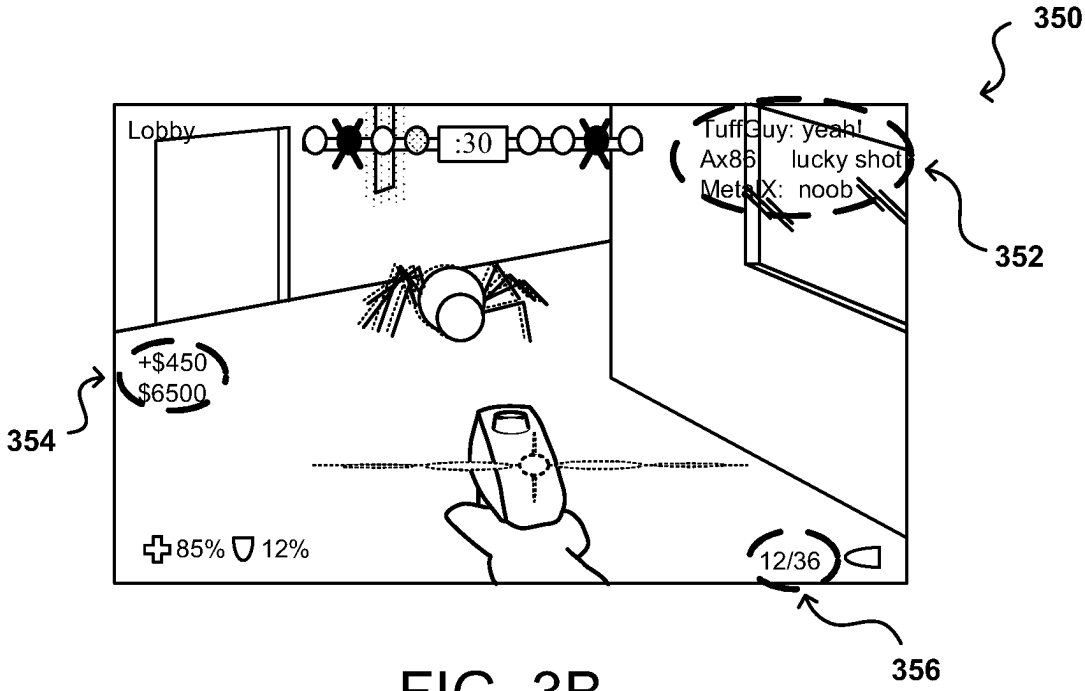

350
Lobby
:30
TuffGuy: yeah!
Ax86 lucky shot
MetaX: noob
352
+$450
$6500
354
85% 12%
12/36
356

FIG. 3B

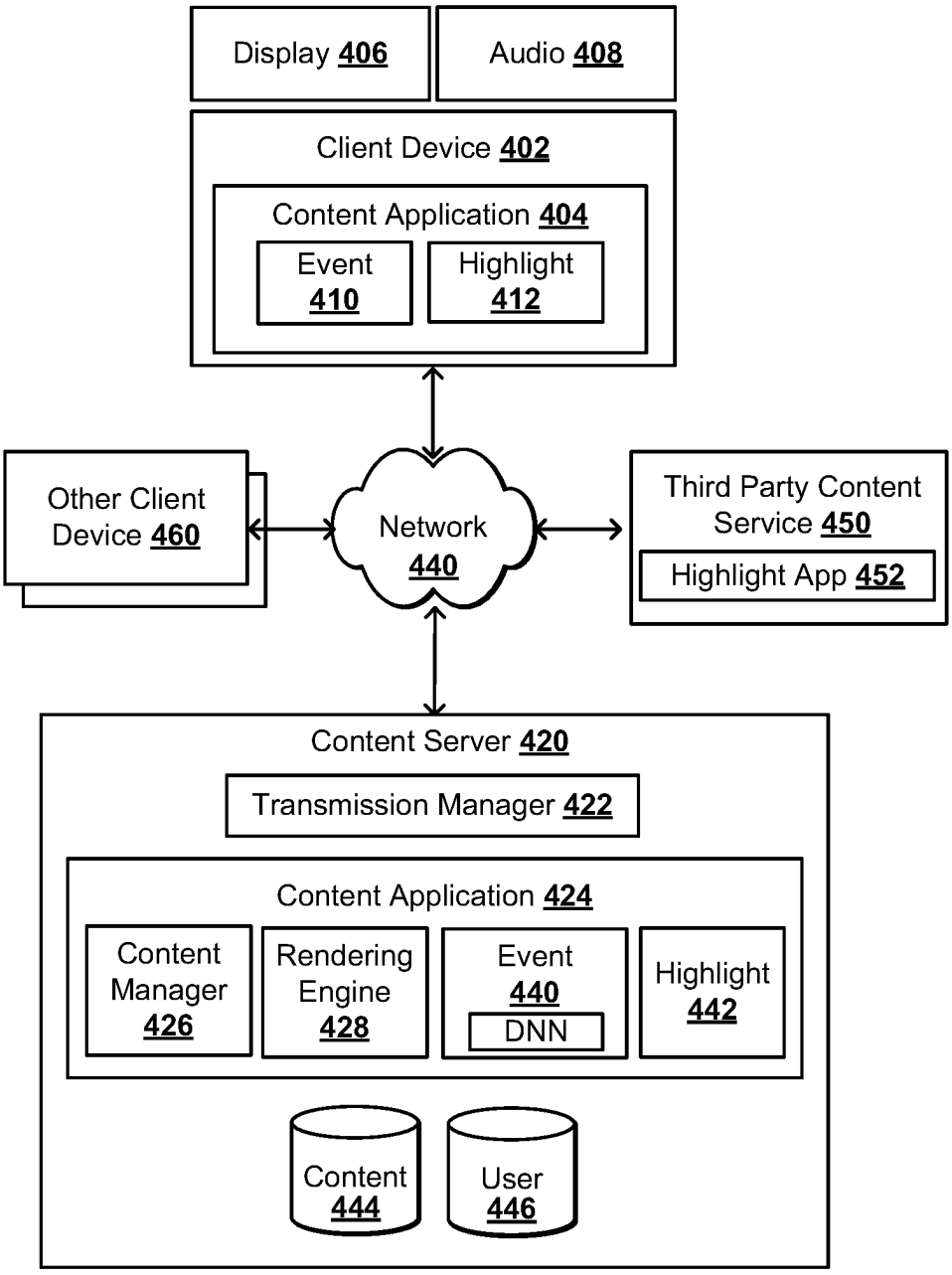

Display 406
Audio 408
Client Device 402
Content Application 404
Event 410
Highlight 412
Other Client Device 460
Network 440
Third Party Content Service 450
Highlight App 452
Content Server 420
Transmission Manager 422
Content Application 424
Content Manager 426
Rendering Engine 428
Event 440
DNN
Highlight 442
Content 444
User 446

FIG. 4

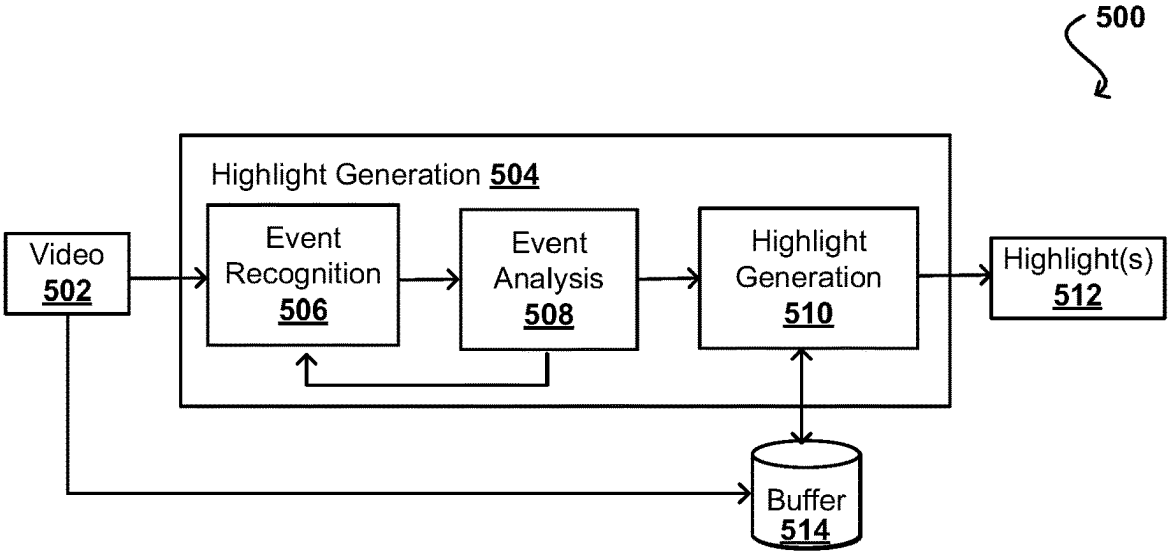

500
Highlight Generation 504
Video 502
Event Recognition 506
Event Analysis 508
Highlight Generation 510
Highlight(s) 512
Buffer 514

FIG. 5A

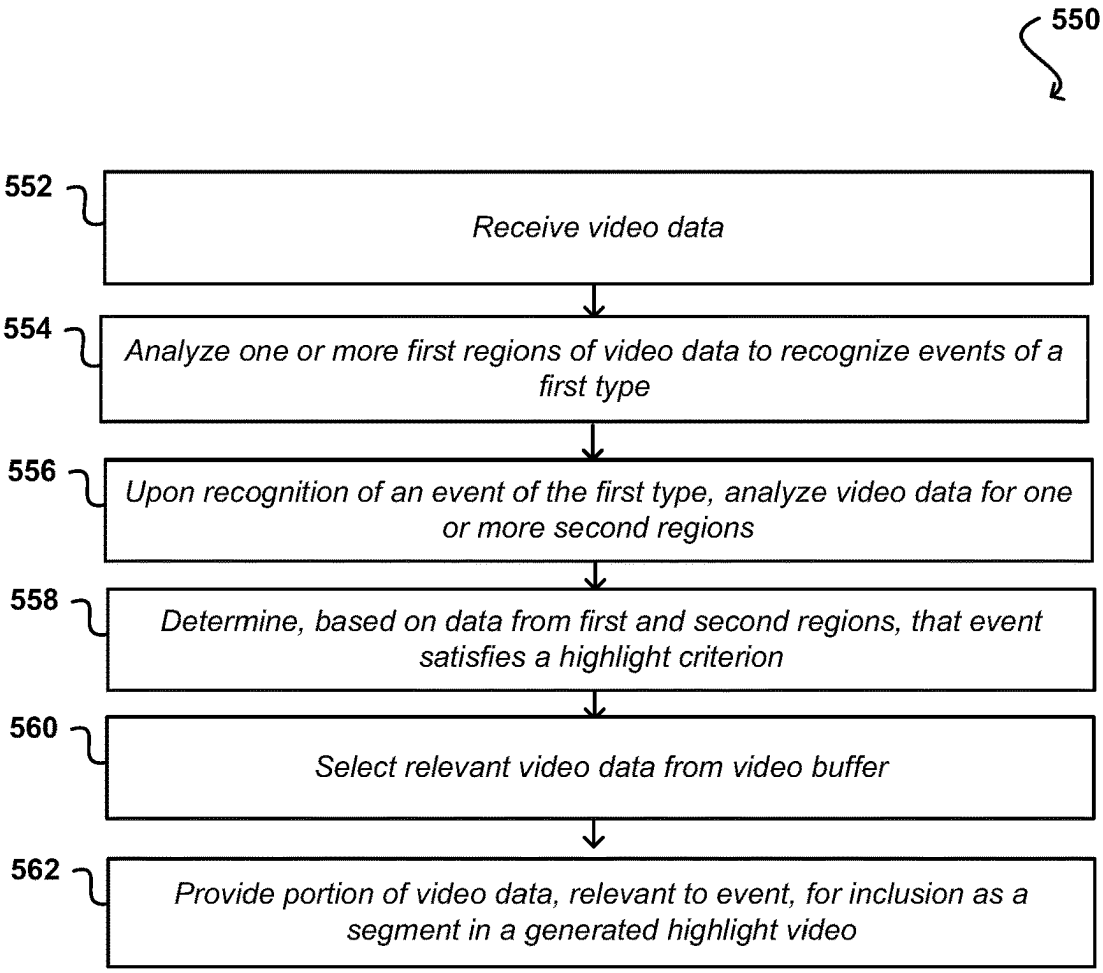

550
552
Receive video data
554
Analyze one or more first regions of video data to recognize events of a first type
556
Upon recognition of an event of the first type, analyze video data for one or more second regions
558
Determine, based on data from first and second regions, that event satisfies a highlight criterion
560
Select relevant video data from video buffer
562
Provide portion of video data, relevant to event, for inclusion as a segment in a generated highlight video

FIG. 5B

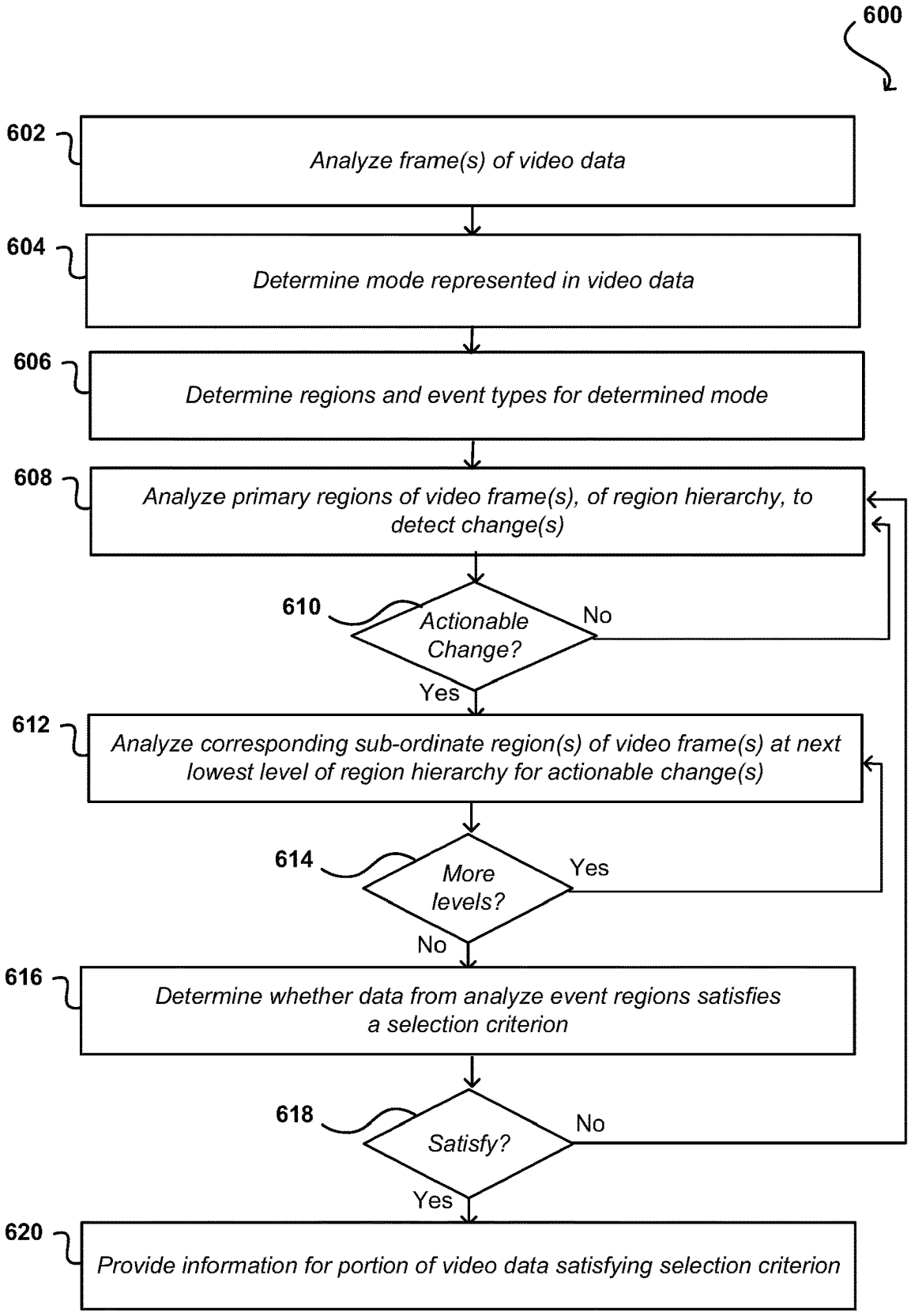

600
602
Analyze frame(s) of video data
604
Determine mode represented in video data
606
Determine regions and event types for determined mode
608
Analyze primary regions of video frame(s), of region hierarchy, to detect change(s)
610
Actionable Change?
No
Yes
612
Analyze corresponding sub-ordinate region(s) of video frame(s) at next lowest level of region hierarchy for actionable change(s)
614
More levels?
Yes
No
616
Determine whether data from analyze event regions satisfies a selection criterion
618
Satisfy?
No
Yes
620
Provide information for portion of video data satisfying selection criterion

FIG. 6

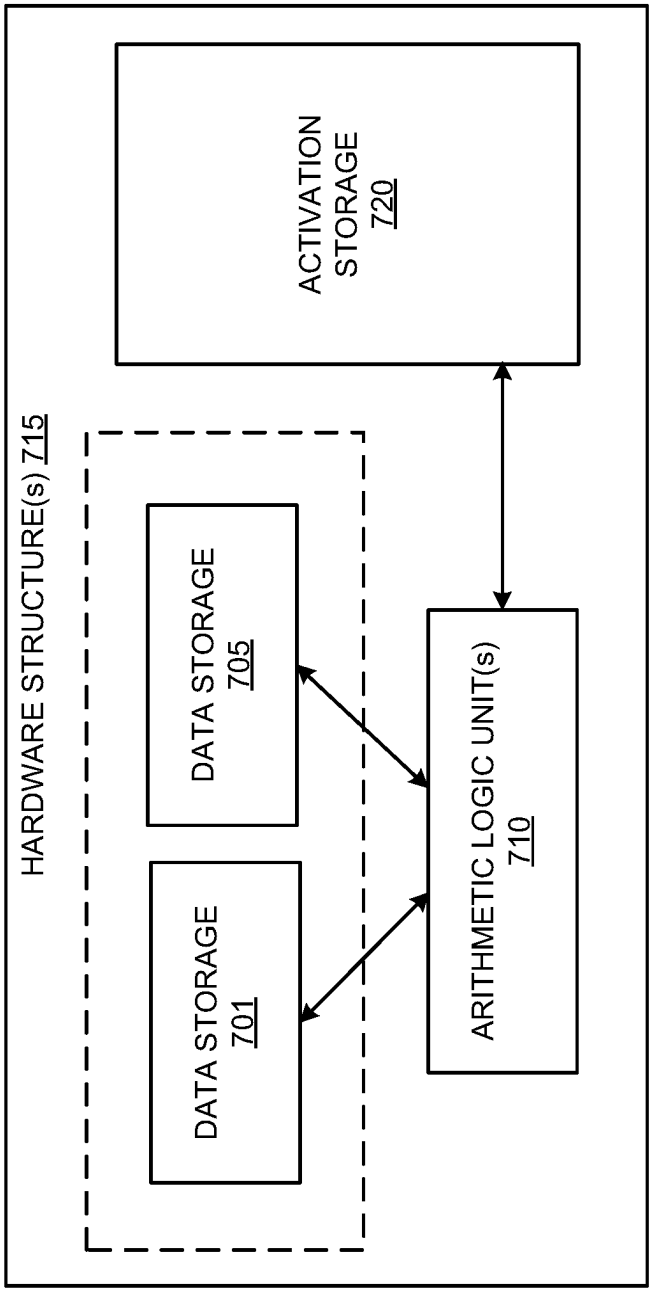

ACTIVATION STORAGE 720
HARDWARE STRUCTURE(s) 715
DATA STORAGE 705
DATA STORAGE 701
ARITHMETIC LOGIC UNIT(s) 710

FIG. 7A

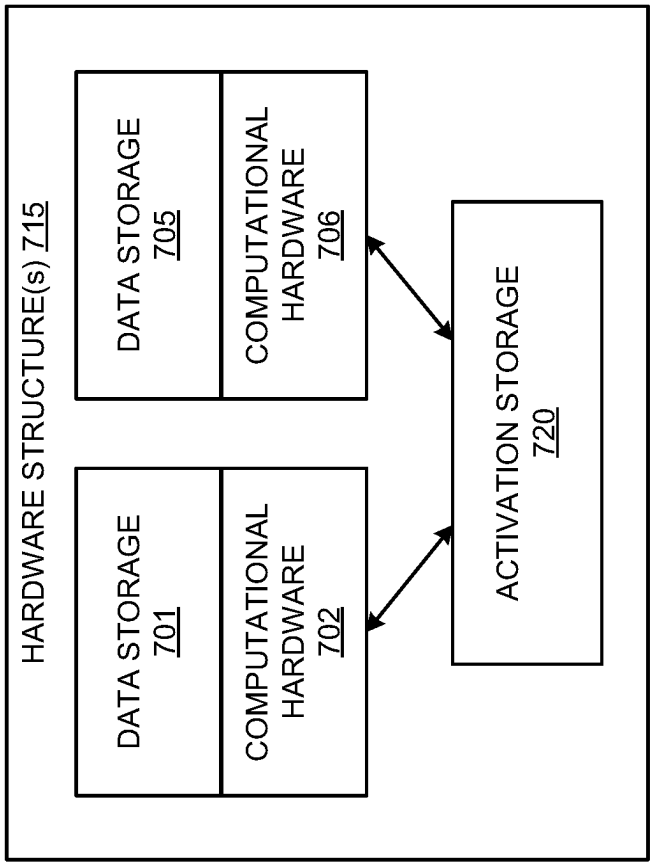

HARDWARE STRUCTURE(s) 715
DATA STORAGE 701
COMPUTATIONAL HARDWARE 702
DATA STORAGE 705
COMPUTATIONAL HARDWARE 706
ACTIVATION STORAGE 720

FIG. 7B

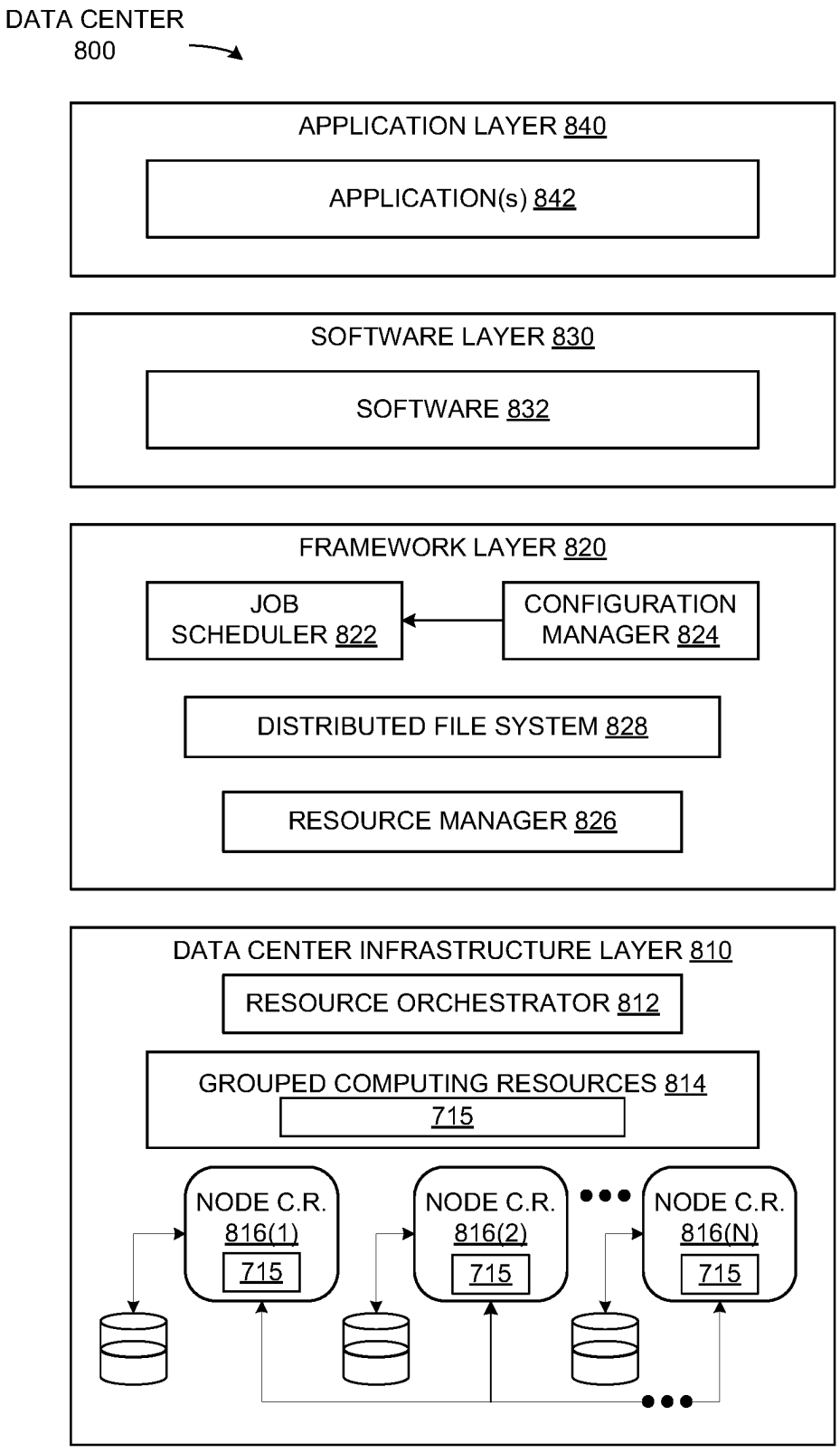

DATA CENTER
800
APPLICATION LAYER 840
APPLICATION(s) 842
SOFTWARE LAYER 830
SOFTWARE 832
FRAMEWORK LAYER 820
JOB SCHEDULER 822
CONFIGURATION MANAGER 824
DISTRIBUTED FILE SYSTEM 828
RESOURCE MANAGER 826
DATA CENTER INFRASTRUCTURE LAYER 810
RESOURCE ORCHESTRATOR 812
GROUPED COMPUTING RESOURCES 814
715
NODE C.R. 816(1)
715
NODE C.R. 816(2)
715
NODE C.R. 816(N)
715

FIG. 8

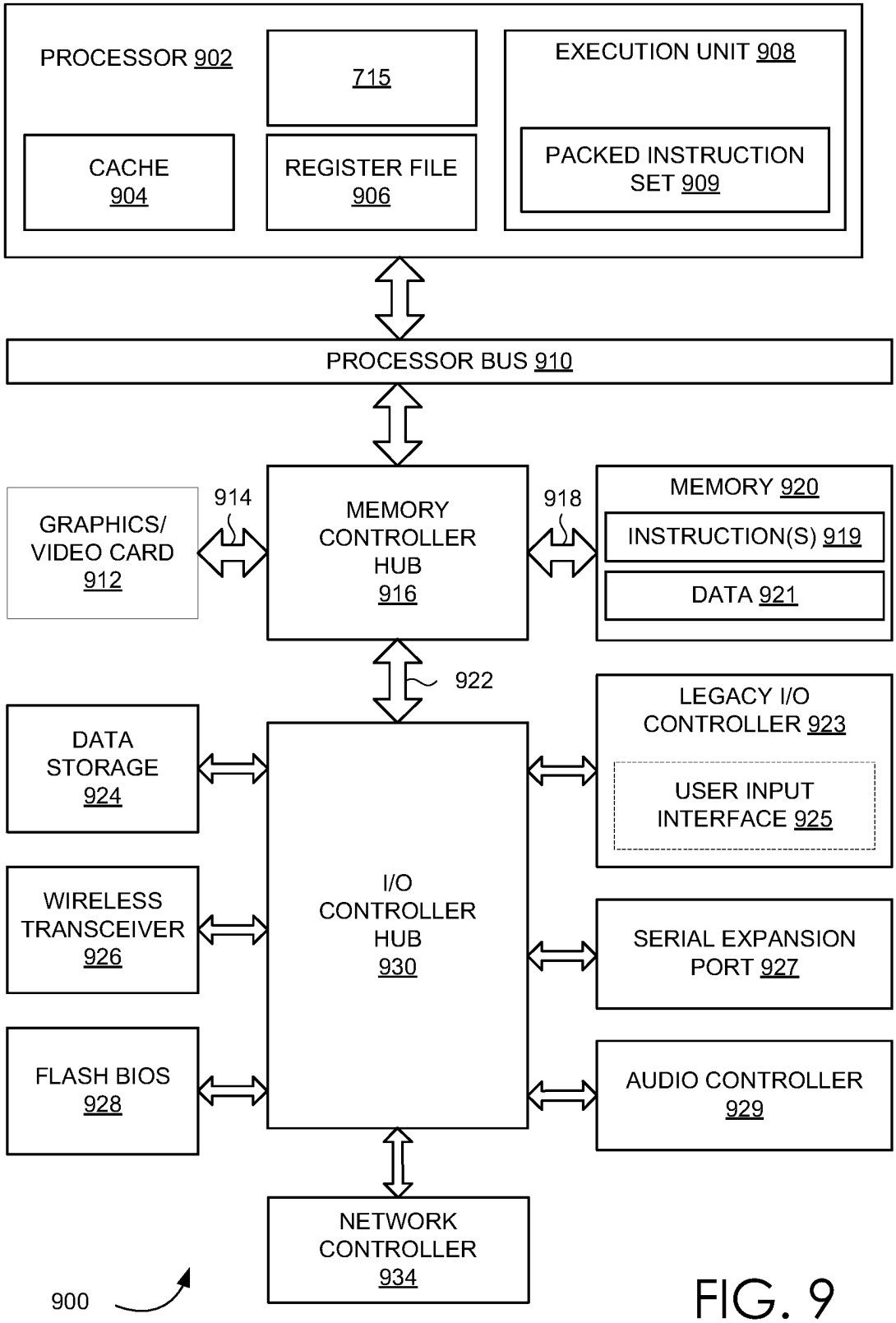

PROCESSOR 902
715
EXECUTION UNIT 908
CACHE 904
REGISTER FILE 906
PACKED INSTRUCTION SET 909
PROCESSOR BUS 910
GRAPHICS/ VIDEO CARD 912
914
MEMORY CONTROLLER HUB 916
918
MEMORY 920
INSTRUCTION(S) 919
DATA 921
922
LEGACY I/O CONTROLLER 923
USER INPUT INTERFACE 925
DATA STORAGE 924
WIRELESS TRANSCEIVER 926
I/O CONTROLLER HUB 930
SERIAL EXPANSION PORT 927
FLASH BIOS 928
AUDIO CONTROLLER 929
NETWORK CONTROLLER 934
900

FIG. 9

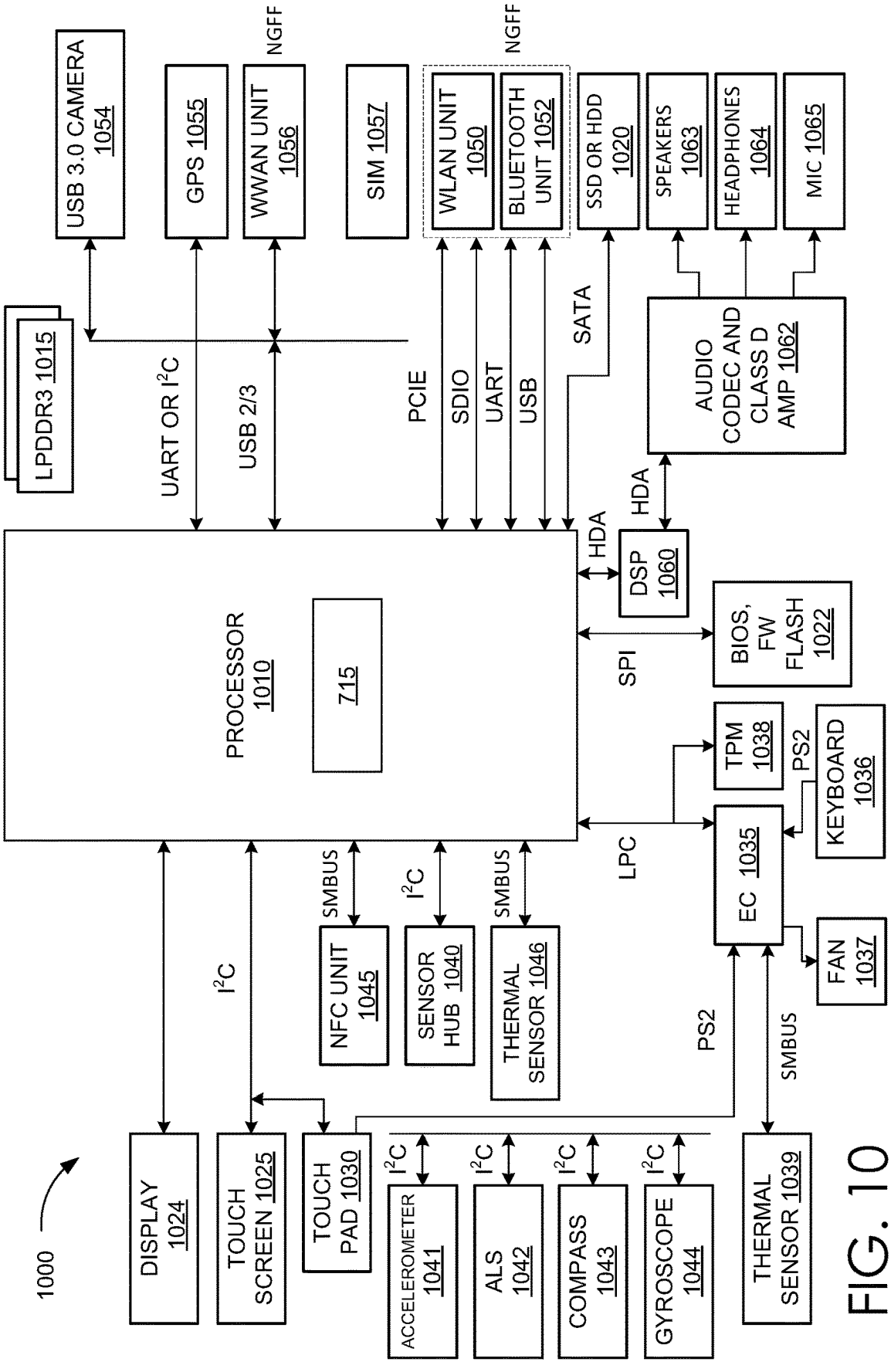

1000
DISPLAY 1024
TOUCH SCREEN 1025
TOUCH PAD 1030
ACCELEROMETER 1041
ALS 1042
COMPASS 1043
GYROSCOPE 1044
THERMAL SENSOR 1039
NFC UNIT 1045
SENSOR HUB 1040
THERMAL SENSOR 1046
FAN 1037
EC 1035
TPM 1038
KEYBOARD 1036
BIOS, FW FLASH 1022
DSP 1060
PROCESSOR 1010
Z15
LPDDR3 1015
USB 3.0 CAMERA 1054
GPS 1055
WWAN UNIT 1056
NGFF
SIM 1057
WLAN UNIT 1050
BLUETOOTH UNIT 1052
NGFF
SSD OR HDD 1020
SPEAKERS 1063
HEADPHONES 1064
MIC 1065
AUDIO CODEC AND CLASS D AMP 1062
UART OR I2C
USB 2/3
PCIE
SDIO
UART
USB
SATA
HDA
HDA
SPI
LPC
PS2
PS2
SMBUS
I2C
SMBUS
I2C
SMBUS
I2C
I2C
I2C
I2C

FIG. 10

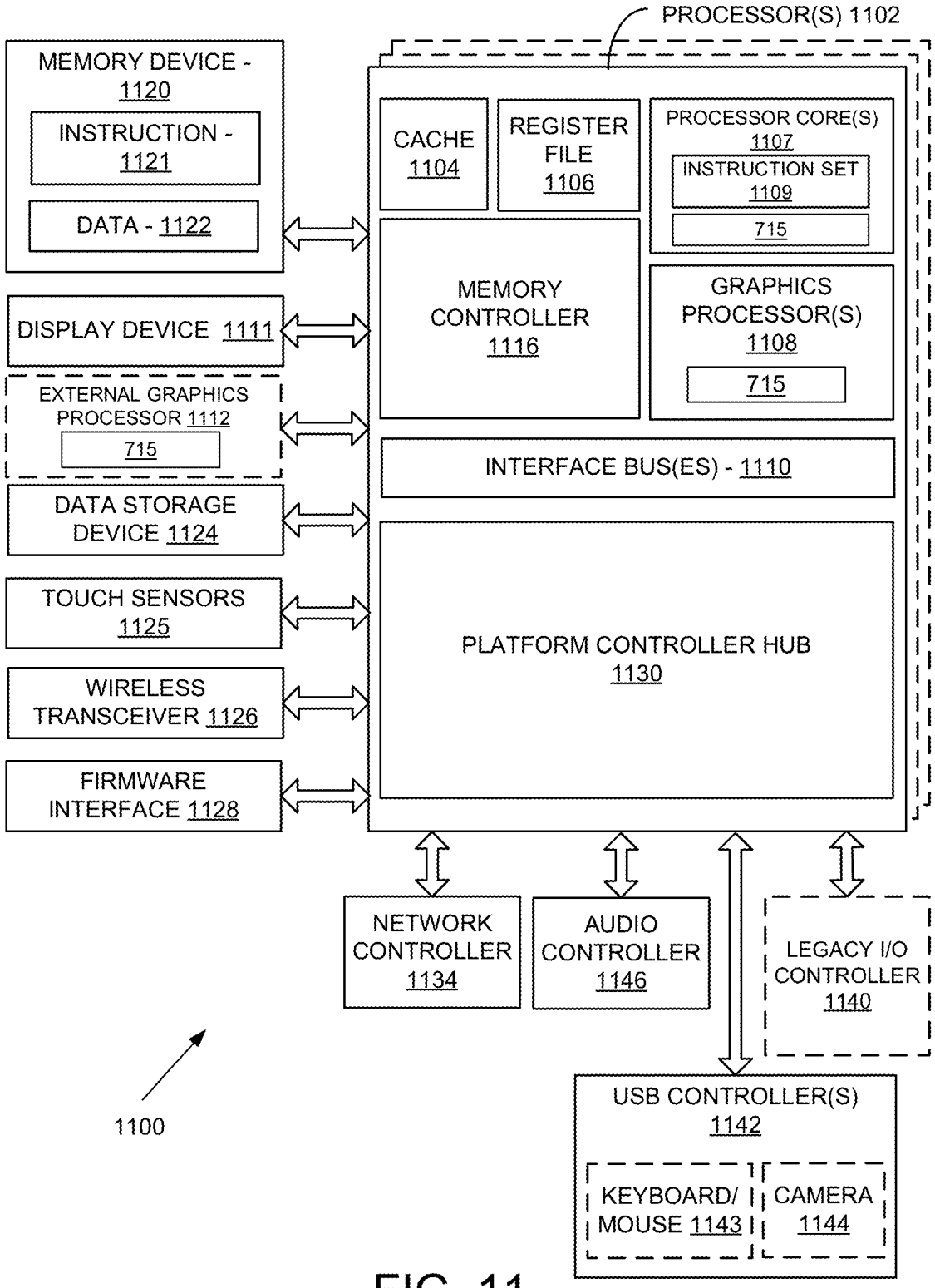

PROCESSOR(S) 1102
MEMORY DEVICE - 1120
INSTRUCTION - 1121
DATA - 1122
CACHE 1104
REGISTER FILE 1106
PROCESSOR CORE(S) 1107
INSTRUCTION SET 1109
715
MEMORY CONTROLLER 1116
GRAPHICS PROCESSOR(S) 1108
715
DISPLAY DEVICE 1111
EXTERNAL GRAPHICS PROCESSOR 1112
715
INTERFACE BUS(ES) - 1110
DATA STORAGE DEVICE 1124
TOUCH SENSORS 1125
PLATFORM CONTROLLER HUB 1130
WIRELESS TRANSCEIVER 1126
FIRMWARE INTERFACE 1128
NETWORK CONTROLLER 1134
AUDIO CONTROLLER 1146
LEGACY I/O CONTROLLER 1140
USB CONTROLLER(S) 1142
KEYBOARD/ MOUSE 1143
CAMERA 1144
1100

FIG. 11

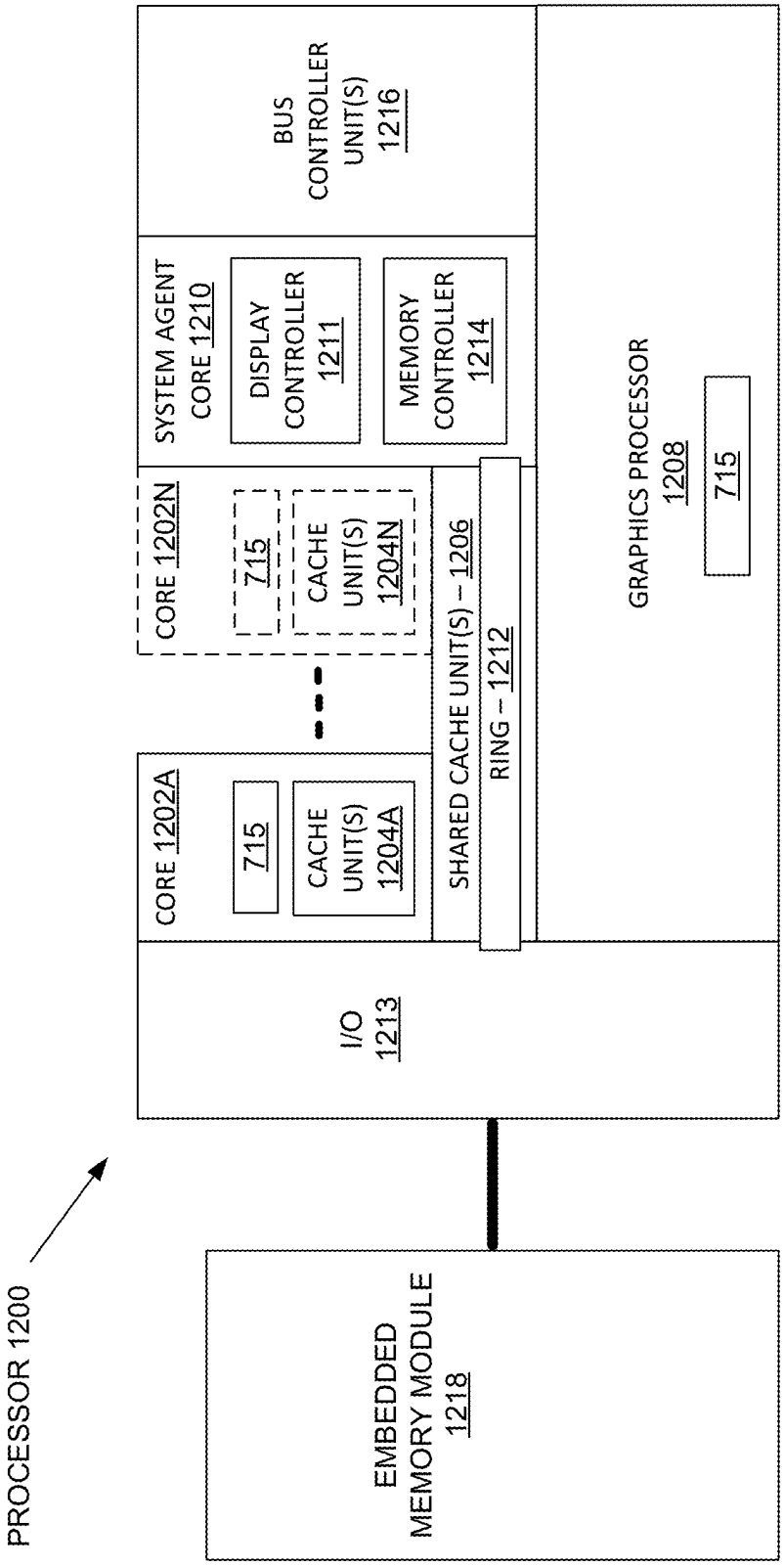

PROCESSOR 1200
EMBEDDED MEMORY MODULE 1218
I/O 1213
CORE 1202A
715
CACHE UNIT(S) 1204A
CORE 1202N
715
CACHE UNIT(S) 1204N
SHARED CACHE UNIT(S) – 1206
RING – 1212
SYSTEM AGENT CORE 1210
DISPLAY CONTROLLER 1211
MEMORY CONTROLLER 1214
BUS CONTROLLER UNIT(S) 1216
GRAPHICS PROCESSOR 1208
715

FIG. 12

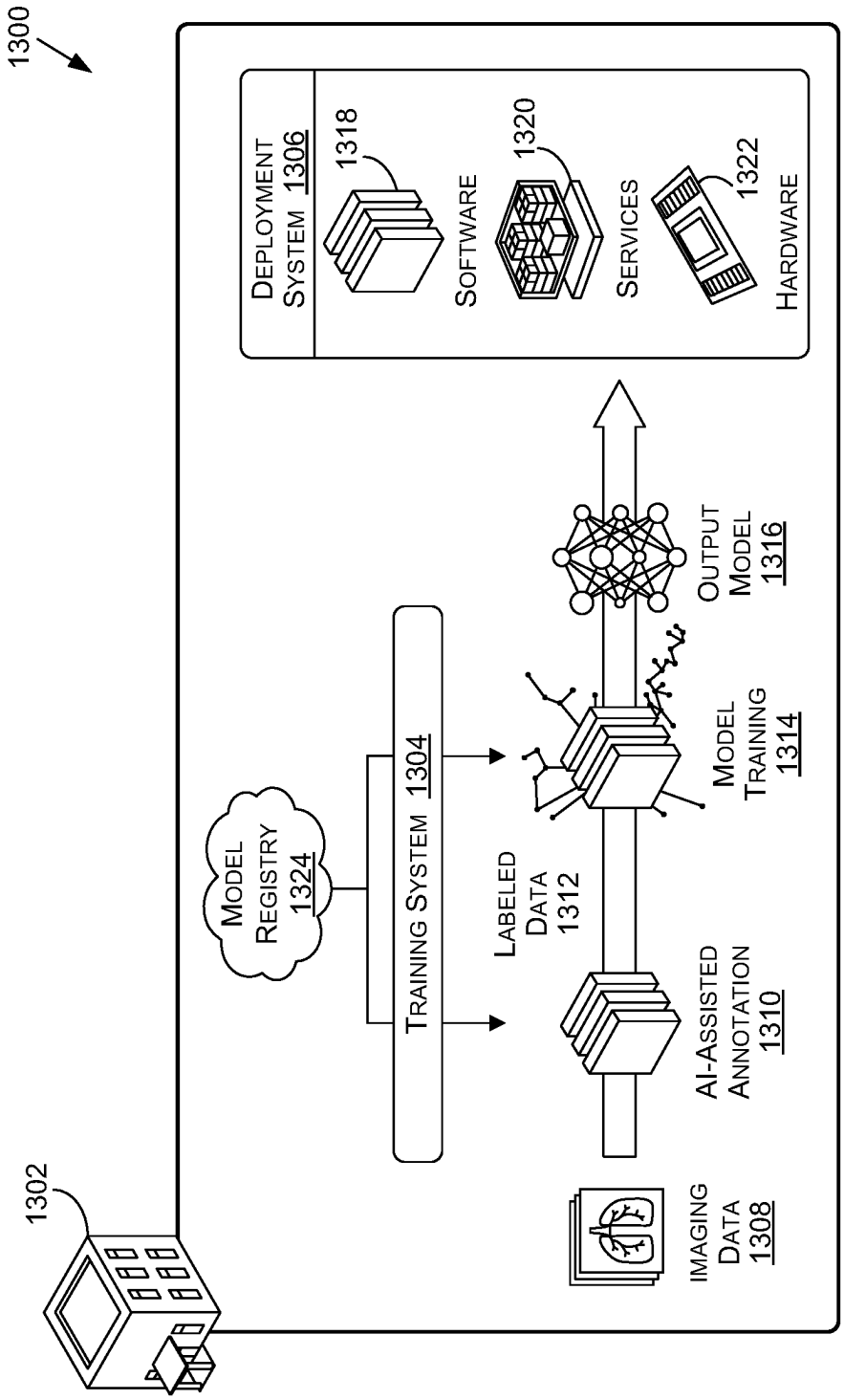

1300
1302
IMAGING DATA 1308
AI-ASSISTED ANNOTATION 1310
LABELED DATA 1312
MODEL TRAINING 1314
OUTPUT MODEL 1316
TRAINING SYSTEM 1304
MODEL REGISTRY 1324
DEPLOYMENT SYSTEM 1306
1318
SOFTWARE
1320
SERVICES
1322
HARDWARE

FIG. 13

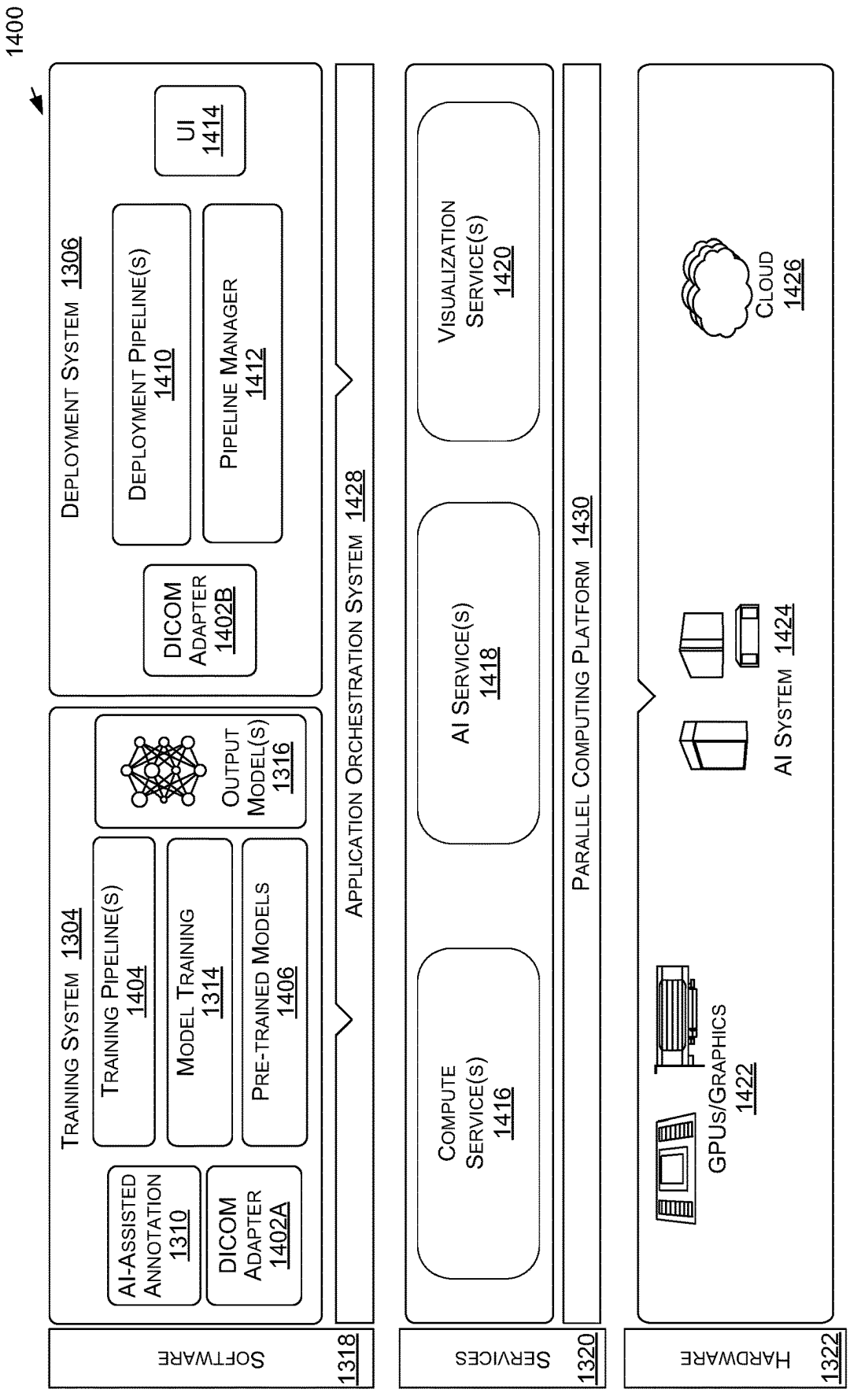

1400
SOFTWARE
1318
TRAINING SYSTEM 1304
AI-ASSISTED ANNOTATION 1310
DICOM ADAPTER 1402A
TRAINING PIPELINE(S) 1404
MODEL TRAINING 1314
PRE-TRAINED MODELS 1406
OUTPUT MODEL(S) 1316
APPLICATION ORCHESTRATION SYSTEM 1428
DEPLOYMENT SYSTEM 1306
DICOM ADAPTER 1402B
DEPLOYMENT PIPELINE(S) 1410
PIPELINE MANAGER 1412
UI 1414
SERVICES
1320
COMPUTE SERVICE(S) 1416
AI SERVICE(S) 1418
VISUALIZATION SERVICE(S) 1420
PARALLEL COMPUTING PLATFORM 1430
HARDWARE
1322
GPUS/GRAPHICS 1422
AI SYSTEM 1424
CLOUD 1426

FIG. 14

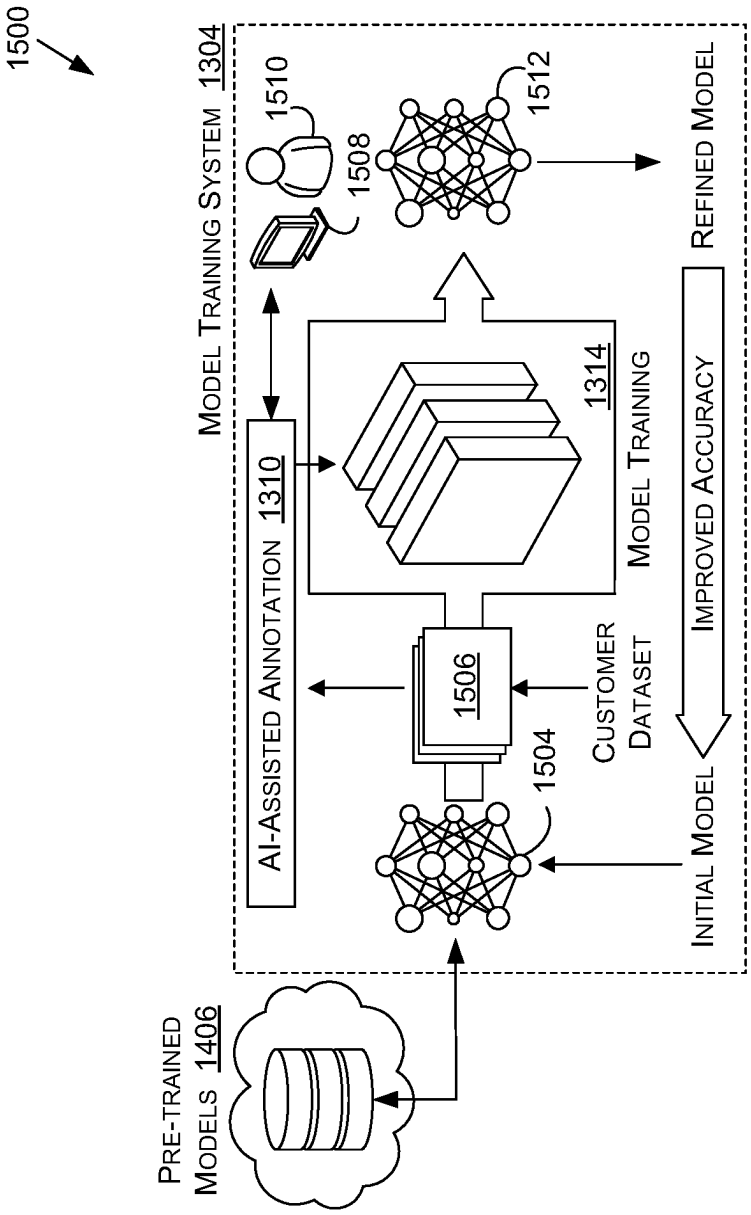

1500
MODEL TRAINING SYSTEM 1304
1510
1508
1512
REFINED MODEL
AI-ASSISTED ANNOTATION 1310
1314
MODEL TRAINING
IMPROVED ACCURACY
1506
CUSTOMER DATASET
1504
INITIAL MODEL
PRE-TRAINED MODELS 1406

FIG. 15A

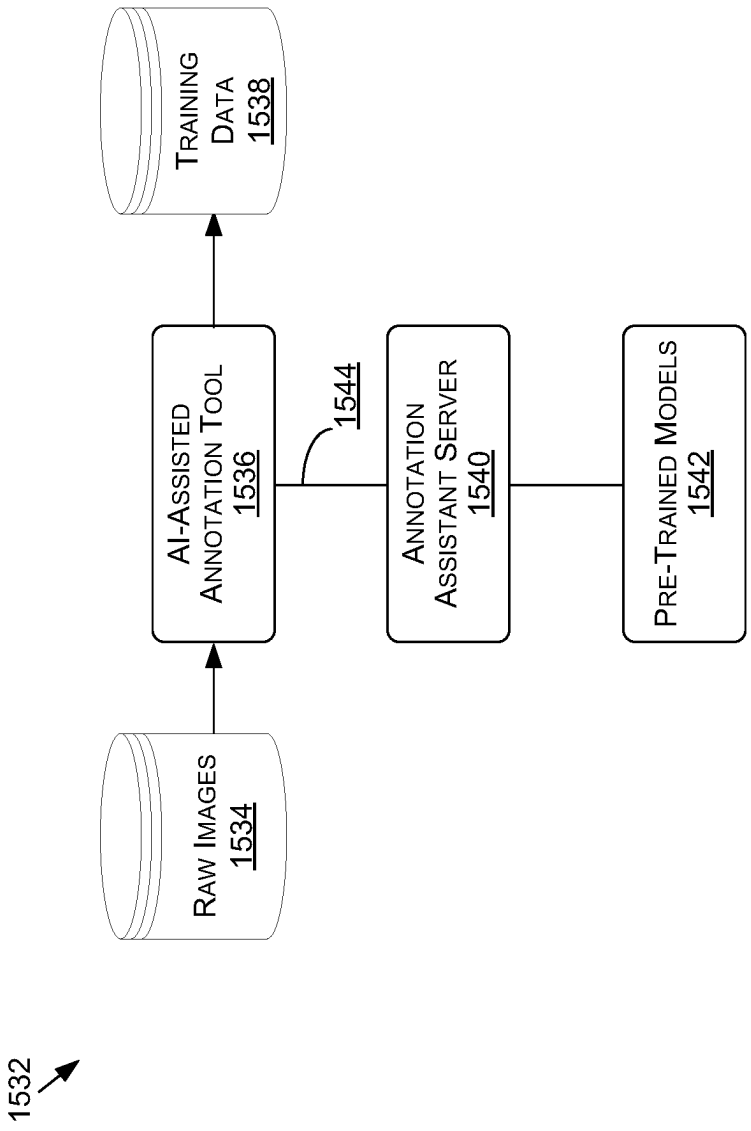

1532
RAW IMAGES
1534
AI-ASSISTED ANNOTATION TOOL
1536
1544
ANNOTATION ASSISTANT SERVER
1540
PRE-TRAINED MODELS
1542
TRAINING DATA
1538

FIG. 15B

GAME EVENT RECOGNITION FOR USER GENERATED CONTENT

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation application and claims priority to U.S. patent application Ser. No. 17/075,377, filed Oct. 20, 2020, entitled "GAME EVENT RECOGNITION FOR USER GENERATED CONTENT", the full disclosure of which is hereby incorporated herein by reference in its entirety for all purposes.

BACKGROUND

Digital content available to end users is continually increasing in complexity and image quality. Content such as video games also comes with increasing types of gameplay available to players, as well as different types of experiences, such as video streaming for non-players and tournament access. Accordingly, approaches to analyzing such content have become more complicated as well, which can prove challenging for devices with limited capacity or where maximum latency requirements can come into play.

BRIEF DESCRIPTION OF THE DRAWINGS

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

FIGS. **1**A and **1**B illustrate images of gameplay, according to at least one embodiment;

FIGS. **2**A, **2**B, **2**C, **2**D, and **2**E illustrate region hierarchies that can be utilized, according to at least one embodiment;

FIGS. **3**A and **3**B illustrate primary and subordinate regions that can be analyzed for an event, according to at least one embodiment;

FIG. **4** illustrates components of an example architecture that can be used to implement aspects of at least one embodiment;

FIGS. **5**A and **5**B illustrates a pipeline and process for generating highlights for input video, according to at least one embodiment;

FIG. **6** illustrates a process for identifying events represented in video that satisfy at least one selection criterion, according to at least one embodiment;

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

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

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

FIG. **9** illustrates a computer system, according to at least one embodiment;

FIG. **10** illustrates a computer system, according to at least one embodiment;

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

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

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

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

FIGS. **15**A and **15**B 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

Approaches in accordance with various embodiments can identify various events or occurrences in media content. This content can include any appropriate type of media content, such as may include audio, video, or image content presented as part of a video, audio, video game, virtual reality (VR), augmented reality (AR), captured performance, or other such experience. In at least one embodiment, this media content can include audio and video representative of one of these other types of experiences, such as streaming video of a gaming session of another player. In at least one embodiment, the types of events or occurrences may depend at least in part upon the type of experience, such as for a gaming experience versus a VR experience. In at least one embodiment, the type of event may also depend on a specific instance of that type of event, such as a specific game being played for a gaming experience.

For example, FIG. **1**A illustrates an image, or frame of video content, corresponding to gameplay of a particular user. In this example, the game is a first person shooter (FPS), or at least a game with an FPS mode, in which a player moves a virtual player through a virtual world to attempt to perform various tasks, which often involves the elimination of one or more enemies, characters, non-player characters (NPCs), or other players. There may be many events or occurrences during a session of such a game, as may involve a player killing an enemy, completing a level, collecting an item, or completing a puzzle. There may be a number of reasons one might want to identify these events, such as to generate player statistics, generate a highlight video, generate a training video, determine player skill level, and so on. In instances where this functionality is generated from within the game, or has at least some integration with a game engine or game server, this information can be provided from the game itself. In other instances, however, this information may not be available from the game and must be determined using only output of the game, such as audio, video, and/or control feedback provided by, or for, the game. In some instances, this may take the form of a gaming platform or video streaming service that may have access to audio and video content for gameplay. That platform or service may want to provide highlight videos, training videos, gaming montages, or other forms of content that are generated from video of game content. In order to accomplish such tasks, this platform, service, or other entity may need to be able to determine events or occurrences represented by that content that are noteworthy or potentially of interest to one or more end users.

One approach to determining events of interest is to analyze the individual frames of video content. In at least one embodiment, this can involve analyzing all content in an image to attempt to identify objects, occurrences, events, actions, or other things of interest that are represented in this content, which hereinafter will be referred to as events for simplicity, although such usage is not intended to limit to only events or limit the interpretation of an event to only these examples. This analysis may include, for example, analyzing the image, audio, and/or video content using one or more neural networks to attempt to recognize or infer any of these events. As illustrated in image **100** of FIG. **1**,

however, there may be many different objects in an image that may change between frames, such that analyzing and tracking all this content over time may be too resource intensive or may come with too much latency for at least some applications.

In at least one embodiment, there may be certain regions of an image that correspond to specific types of information associated with one or more events of interest, such that the complexity of analysis can be reduced by limiting at least some of the analysis to these areas, and attempting to detect or identify certain states or changes in states (e.g., transitions) of content in those regions. For example, it might be desirable to know when a player eliminates another player for purposes of generating a highlight video or montage. Video or image data rendered for the game may include one or more user interface elements **102**, **106** that indicate when a player in a game session is eliminated. There may be various other regions that correspond to graphical user interface (GUI) or heads-up display (HUD) information as well, which may be useful in identifying these and other types of events that occur during gameplay. For example, image **100** includes regions that correspond to various UI elements, as relate to time remaining **104**, in-game chat messages **108**, type of ammunition or weapon selected **110**, amount of ammo remaining, shield **114**, health **116**, virtual player cash **118**, and location **120**. There may be other regions associated with information that only appears at certain times, such as when a player dies and is spectating gameplay of another player. The information contained in at least some of these regions can change over time, and those changes can be indicative of various types of events. In at least one embodiment, events can be determined by detecting changes in one or more of these regions, and combining information for that change with information in one or more other reasons that may be used to determine a type of event that has occurred. For example, if a user element **102** indicates that a player has been eliminated, a player's cash **118** goes up by an amount associated with a kill, an amount of player ammunition **112** went down, and a chat message **108** indicates that a current player killed that other player, then a determination can be made with high certainty that this player eliminated that other player, even if the actual elimination (e.g., the shot by the current player that killed the other player's avatar or character) was not detected or not analyzed in the video data. If an element appears in a region to show that a player is simply spectating at a specific time, then any kill that occurred at that time was not initiated by the player and then therefore may not qualify to be selected for a highlight, depending at least in part upon the relevant event rule or selection criteria. Various other actions or events can be determined as well, as may relate to a player skydiving, achieving a higher level, or performing another action or accomplishment that may be worthy of inclusion as a highlight.

In at least one embodiment, information in each of these regions can be analyzed and/or evaluated for each video frame in order to accurately detect game events. Such a brute force approach can be relatively resource intensive, however, particularly for a large number of events of complicated elements that may be contained in those regions. For example, in many cases a UI element will be overlaid on varying gameplay elements over time, and it may take some amount of segmentation or image recognition to determine those elements for different frames.

Approaches in accordance with various embodiments can take advantage of the fact that there may be specific regions that are highly indicative of the occurrence of an event, or that will change or have a specific state or value corresponding to an event of interest (although these elements may change for other events as well). For example, if player eliminations are to be used to select highlights for a video, then a UI element **102**, **106** that updates each time a player is eliminated can be a primary indicator of the occurrence of this event. If a player elimination icon is not updated or does not undergo a change in state, then there is no reason to evaluate other information in the image to determine whether a current player eliminated another player.

It may be the case that highlights are not being generated for an entire game session, or all players therein, but may be generated for a specific player, and is to include only highlights that are relevant to that player. In such a situation, the changing of the player elimination UI element **102** may be insufficient to identify an event where a current player eliminated another player, as there might have been another reason for that other player being eliminated, such as by falling off a level or being killed by a different player. Accordingly, it may be necessary to evaluate information in these other regions as well. In at least one embodiment, events of interest may therefore have a primary region identified that is indicative of a type of event occurring, after which information in these other regions can be analyzed, such as may be part of a multi-pass process. In this way, many of these "subordinate" regions (or child regions in a region hierarchy) then are only analyzed if a state or value of an icon, text, or other UI element in a corresponding primary region has changed or otherwise had a specific state presented. In at least some embodiments, the subordinate regions that are evaluated for a specific type of event may include only those that are determined to be relevant to that type of event, as may be determined using one or more rules generated, customized, or otherwise provided or obtained for that type or instance of content. In at least one embodiment, these subordinate regions can also have parent-child relationships among them.

FIG. **1**B illustrates another example image **150** corresponding to a frame of gameplay. As illustrated, this image contains various objects, as well as a number of UI elements. In at least one embodiment, at least some of these UI elements can be assigned to primary or subordinate regions that can be used to identify specific types of events or occurrences. In this example, the player is driving a vehicle in a racing game, or at least a racing mode in a game session that may include multiple different modes of gameplay. There may be multiple events of interest in such a game, such as a player winning a race, taking the lead, or wrecking another player. For each of these types of events, there may be a primary region identified that is indicative of that type of event. For example, this display includes regions for UI elements relating to time remaining **152**, players eliminated **154**, mode of operation **156**, speed **158**, engine load **160**, location **162**, leaderboard **164**, position **166**, and score **168**. For each type of event, there may be a rule indicating which region is a primary region, and which regions are subregions. As will be discussed in more detail later herein, these rules can also specify subordinate regions of a region or event hierarchy, where those regions are only evaluated in response to a state, or change in state, of at least one region at a higher level in that hierarchy.

For an event where a player passes into first place, a primary region may be the place indicator **166** and/or the leaderboard **164**. While a position region **166** may be enough to indicate that a player has entered first place, that may have resulted from other players dropping out of the race or the player being the only human player racing at the

current time, which may be determinable in conjunction with the leaderboard **164** or player elimination icons **154**. A map **162** can also be used to determine proximity of other vehicles, which can be used to determine whether an event is highlight worth, such as where there are other vehicles nearby, and preferably just behind a current player's vehicle. Thus, at least these regions may be evaluated to determine whether the player entering into first qualifies for a highlight by satisfying at least one highlight selection criterion. For example, a player passing the first place car may qualify, but the player entering first place because the other player drops out of the race may not qualify for highlight selection. As with the prior example, an elimination event may use an elimination icon **154** as a primary region, with other subordinate regions evaluated to determine whether a current player was responsible for that elimination (or whether that elimination otherwise satisfies a criterion for highlight inclusion). Winning a race may be determined using a primary region that indicates victory or place, but information in other regions such as other players still playing or having time left on a clock may be necessary to determine whether to include this event in a highlight. As will be discussed in more detail later herein, selection of an event for inclusion in a highlight video may include pulling at least some video before and after event detection from a buffer for inclusion in that highlight video. In other embodiments, a time stamp can be stored for that highlight, along with event information such as type of event, and that information can be used to extract relevant portions of that video at a later time for dynamic highlight video generation, such as where a viewer wants to see only a certain type of highlight, such as only kills, takings of the lead, or victories.

FIG. **2**A illustrates a set of example regions **200** that can be identified for a given game, or mode of gameplay within a game. In at least one embodiment, these fields or regions can be selected or customized specifically for a game, game mode, or type of game. As mentioned, each of these fields or regions may correspond to a specific type of information located in a specific region, or locatable region, in an image or video frame of gameplay, where that information may be represented by text, an icon, or another graphical object or element. In at least one embodiment, at least one audio region may be specified as well, as may relate to a sound or music that plays in response to, or along with, a type of event. Other output, such as haptic feedback, may be analyzed if that information is available. In this example, each of these regions can be treated similarly, such that they can all be evaluated concurrently for each frame using a brute force method, or for at least a subset of frames in a video, such as every third frame if it is desirable to reduce resource requirements while still able to retain event detection accuracy.

As mentioned, however, there may be at least some regions that only need to be evaluated for an event if a state of a primary field or region has changed. As an example, the regions in FIG. **2**B have been divided into two levels or layers of an event hierarchy **220**, where each level corresponds to a different state and can be evaluated in a separate analysis pass. In this example, a game has a rule for a "kill" type of event, where a kill icon is designated as a primary region, and regions high kill and bot mode are identified as subordinate regions. These subordinate regions will only be evaluated for frames in which, or proximate which, a kill icon changes or has a determined state or value. If a kill icon does not change or have one of these values, then these subordinate regions will not be evaluated. Other regions, such as flashbang and spectator band, may be evaluated on each frame, or may not be evaluated, but may not be included in the event rule. In the hierarchy **240** of FIG. **240**, however, a rule may specify a primary region, such as kill icon, and all other regions then become a subordinate region for at least that rule, and are then checked, analyzed, or evaluated only when a kill icon reaches, changes, or represents a specific state or value. In the example hierarchy **260** of FIG. **2**D, there may be additional levels in such a hierarchy, where certain fields or regions are only analyzed if at least one state, change, or value in a higher level of the hierarchy means that, according to the respective rule, that field or region should be checked, analyzed, or evaluated. In at least one embodiment, a game can have any number of fields or regions, and a rule may select any number of these regions to be included at any of a number of different levels of an event hierarchy. The rule can also specify one or more criteria for regions of a lower level to be evaluated, such as a field or region in a higher level having a specific value or state, being within or outside a specified range, changing by more than or less than a threshold amount, and so on.

It may be the case that a given game or experience has multiple modes of operation or gameplay. For example, a game may have a mode or level that operates as a first person shooter, a mode where a player operates a vehicle, a mode where a player must solve a puzzle, and so on. For each of these different modes, there may be different fields or regions displayed that may include different types of information. For each of these modes, there may also be different types of events that are to be selected for a highlight video. Accordingly, in at least one embodiment an event hierarchy might include different rules with different primary regions as illustrated in example hierarchy **280** of FIG. **2**E. In such situations, there may be one or more regions that are evaluated to determine a current mode of gameplay. For that given mode, there may be one or more primary regions to be analyzed to detect types of events relevant for that mode of gameplay. In some embodiments, frames can be analyzed to attempt to determine a presence of one or more regions to assist in determining a current mode of gameplay. In at least one embodiment, determination of a game mode can also cause regions unrelated to that game mode to be filtered, or removed, from consideration. In this way, game rules, events, and regions can be mapped to a tree of text, icons, sounds, or other elements present, as may be part of a GUI or HUD.

FIGS. **3**A and **3**B provide an example of how such a hierarchy can be utilized in accordance with at least one embodiment. In the image **300** of FIG. **3**A, a primary region **302** is illustrated that contains a graphical element of interest, in this case a player status bar that indicates the status of other players in a game. While an oval region is illustrated, it should be understood that the region can have any appropriate size and shape that bounds at least a relevant portion of an element of interest, and may have at least some buffer to allow for slight variation, where a rectangular bounding box may be used in many instances. In an example where player kills are a trigger for a highlight to be generated, the player status bar may be used as a determining trigger in a primary region. This primary region **302** can be analyzed, as part of a first or primary pass, on each frame to attempt to determine when there is a meaningful change in state. In at least one embodiment, this can include the bar changing to illustrate that a player has been eliminated or is no longer active in this current game session. There may be other states as well, such as to indicate when a player has been knocked down or has low health, and these may not satisfy the selection criterion for a highlight in this example.

When an actionable change is detected in this primary region **302**, such as when an icon of the status bar changes to indicate that a player is no longer active in this session, other information for that frame **350**, or at least one proximate frame, can be analyzed during at least one subsequent pass to attempt to determine whether a highlight should be generated, or other such action taken. In this example, there may be three subordinate regions at a lower level in an event hierarchy, under the status bar primary region. In this example, these include a chat region **352**, a cash region **354**, and an ammunition region **356**. These subordinate regions can be analyzed to determine whether an event has occurred that should trigger a highlight, based on a detected change in the primary region. In this example, chat messages in the chat region **352** may be analyzed, such as by using a text analyzer, to attempt to determine whether information is provided as to the type or source of an event, such as indication of a player making a kill. A change in an amount of cash in a cash region **354** may be indicative of a kill if a player receives an amount of cash for a kill, and the cash has recently gone up by that amount. Further, an amount of ammunition in an ammunition region **356** can be analyzed to determine whether that amount recently changed to reflect ammunition being used, as an indication that no ammunition has been used recently may, in at least this game, be an indication that this player did not lead to the death of the other player. Various other types of regions or analysis can be used as well within the scope of the various embodiments. Further, there may be additional subordinate regions at lower levels of an event hierarchy for this event that may be analyzed in response to one or more of these subordinate regions **352**, **354**, **356** having a determined state, or change in state.

In at least one embodiment, at least some of this image or video content may be provided or presented locally on a client device **402** as illustrated in FIG. **4**. At least a portion of this content may be provided by a content server **420**, such as a game server or provider system, across at least one wired or wireless network **440**. In at least one embodiment, content to be presented may include various types of content, as may include video game, virtual reality (VR), augmented reality (AR), mixed reality (MR), image, textual, audio, haptic, or video content. Client device **402** may include or comprise a device such as a desktop computer, notebook computer, gaming console, smart phone, tablet computer, VR headset, AR/MR goggles, a wearable computer, or a smart television.

In some embodiments, content provided to, or generated on, client device **402** may include highlights from specific media, such as a game hosted on client device **402**, content server **420**, or third party content service **450**. In some embodiments, media may be received to client device **402** and highlights determined using an event detector **410** and highlight generator **412** of a content application executing on client device **402**. In other embodiments, an event detection module **440** and highlight generator **442** might run in a content application **424** running on content server **420**, or in a highlight application **452** on a third party content service **450**, where those highlights can then be transmitted to one or more other client devices **460** for display as well. As mentioned, one or more neural networks may be used for purposes such as event detection, criteria evaluation, and/or highlight selection.

In at least one embodiment, client device **402** can generate content for a session, such as a gaming session or video viewing session, using components of a content application **404** on client device **402** and data stored locally on that client device. This content may be analyzed in various embodiments for purposes such as to generate highlights or training videos. In at least one embodiment, a content application **424** (e.g., a gaming or streaming media application) executing on content server **420** can initiate a session associated with at least client device **402**, as may utilize a session manager and user data stored in a user database **446**, and can cause content **444** to be determined by a content manager **426** and rendered using a rendering engine **428**, if needed for this type of content or platform, and transmitted to client device **402** using an appropriate transmission manager **422** to send by download, streaming, or another such transmission channel. In at least one embodiment, client device **402** receiving this content can provide this content to a corresponding content application **404**, which may also or alternatively include a rendering engine **410** for rendering at least some of this content for presentation via client device **402**, such as video content through a display **406** and audio, such as sounds and music, through at least one audio playback device **408**, 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 **402** such that transmission over network **440** 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 **420**, or content database **444**, to client device **402**. 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 **450** that may also include a highlight application **452** for generating or providing content.

FIG. **5**A illustrates components of an example highlight generation system **500** that can be utilized in accordance with at least one embodiment. In this example, video data **502** is received for analysis in selecting highlight clips. This video data can include a full download or transmission of data, or streaming of live data content, among other such options. In this example, the video content is provided as input to a highlight generation module **504**, system, or service. The video can be passed to an event recognition module **506** that can attempt to identify specific events represented in the video. This can include, for example, analyzing content in specific regions of video and determining a state, or change in state, for one or more elements in that region. One or more neural networks may be utilized that are trained to classify different types of objects that may be represented in a video frame. As mentioned, there may hierarchical levels of regions, and an event recognition module might first analyze only content for one or more primary regions. In this example, the event recognition module **506** can analyze information in these primary regions, and can pass this information to an event analysis module **508**. An event recognition module can use one or more event auto-recognition algorithms, processes, or deep learning approaches to recognize events, or objects and occurrences associated with various types of events. This event analysis module can analyze the information to determine whether the information in one or more primary region has a state, or has had a change in state, that warrants further investigation for highlight selection. If so, the event recognition module **506** can evaluate one or more subordinate regions, as may be determined by one or more rules for one or more specific types of event. Information from these subordinate regions can then be passed to the event analysis module **508** to determine whether one or more highlight

selection criteria have been satisfied. For a kill event, a highlight selection criterion might include a determination that a current player killed another player with at least 85% certainty based at least in part upon the information from these regions. If such a criterion is satisfied, information for that event can be passed to a highlight generation module **510**, which can be responsible for generating a corresponding highlight. This can include, for example, pulling video data from a video buffer **514**, where the video may include some amount of video content before, and after, a timing of the event. In another embodiment, this may include determining timing information for this highlight to be used to pull that video content at a later time. This highlight information for one or more highlights **512** can then be provided as output, to be stored for subsequent viewing or presentation via a client device as those highlights are determined.

FIG. **5**B illustrates a process **550** for determining highlights using a system such as that described with respect to FIG. **5**A. It should be understood that for this and other processes presented herein that there can be additional, fewer, or alternative steps performed in similar or alternative order, or at least partially in parallel, within scope of various embodiments unless otherwise specifically stated. As mentioned previously, identifying events can be useful for other purposes as well, such as for testing or training purposes. In this example, video data is received **552** that includes content that may be useful in generating one or more highlights. There may be one or more regions of interest identified for this type of video content, including at least two different types of regions, such as primary and subordinate regions. One or more first regions of this video data can be analyzed **554** to attempt to recognize events of a first type. This may include analyzing primary regions to attempt to identify a state, or change in state, of one or more interface elements. In at least one embodiment, the second regions are not analyzed unless a first type of event is recognized in a video frame for one or the first regions. Upon recognition of an event of the first type, first or second regions of this video data can be analyzed **556**, such as to identify related state information for other interface elements. It may be determined **558**, based at least in part upon data from these first and second regions, that this identified event satisfies a highlight criterion. If so, relevant video data can be selected **560** from an appropriate video buffer or file. At least that portion of the video data, relevant to the determined event, can then be provided **562** for inclusion as a segment in a generated highlight video.

FIG. **6** illustrates an example process **600** for determining whether an event satisfies a selection criterion that can be performed in accordance with at least one embodiment. In this example, one or more frames of video data are analyzed **602**. This can include, for example, attempting to determine one or more objects, actions, events, or occurrences that may be indicative of a game, or mode of gameplay. This data may be determined in at least one embodiment by using one or more neural networks, such as one or more convolutional neural networks (CNNs) trained to recognize different types of objects in image or video data. A mode of gameplay can then be determined **604** based at least in part upon this video data. This may include, for example, a determination as to whether rules should be utilized that relate to driving, sports, or puzzle gameplay of an identified game. Once a current mode is determined, one or more regions and event types can be determined **606** for that game mode of the current game. In at least one embodiment, a customized set of rules and regions can be provided for each game, or type of game, as well as different modes of gameplay or operation within that game. For one or more current frames of video content representative of gameplay, one or more primary regions of a region hierarchy can be analyzed **608** to attempt to address changes, or specific state(s), of elements within those regions. If it is determined **610** that no actionable change has occurred, then the process can continue with one or more subsequent frames.

If it is determined that an actionable change was detected in a primary region, then one or more subordinate regions at a next lowest level of the region hierarchy can be analyzed **612** to attempt to determine actionable changes, transitions, or specific state values. If it is determined **614** that there are more levels in this hierarchy, and such analysis is warranted based on information from regions at a current level, then the process can continue at this next lowest level. Once the relevant subordinate regions have been analyzed, it can be determined **616** whether data from those regions satisfies a selection criterion. If it is determined **618** that such a criterion has not been satisfied, then the process can continue with one or more subsequent frames. If it is determined that a selection criterion has been satisfied, then information for a portion of the video data that satisfies this selection criterion can be provided **620** for use, such as to generate a highlight or training video. This process can then continue for subsequent frames until the end of the video is reached, a maximum number of highlights has been reached, or another such end criterion is met. In some embodiments, there may be at least one subsequent step to determine, from the selected highlights, which highlights to include in a final highlight sequence or montage.

In some embodiments, event regions and region hierarchies can be determined manually. In at least one embodiment, at least some of these regions and hierarchies can be determined automatically, as may be based at least in part upon event rules for a game or other type of content. In at least one embodiment, a Bayesian approach can be used to determine which regions change along with, or in response to, changes in other regions. Based at least in part upon this data, relationships can be learned that can be used to produce hierarchies of event regions. In at least some embodiments, a user may be able to specify certain fields, rules, events, or hierarchies for generating highlights or otherwise performing tasks based at least in part upon detected events. A user may also be able to activate or deactivate highlights for different game modes or types of gameplay, such as where a user only wants to see certain types of highlights. In at least one embodiment, additional fields can be introduced to an event dictionary to indicate associations with event regions, as well as the type of region or position in an event hierarchy. Inclusion of these labels in an event dictionary can ensure that consistent terminology and labeling is utilized across different games or other types of content.

As mentioned, the determination of events using such an approach can provide additional benefit as well. For example, the ability to track player events with little additional computational overhead provides an ability to more accurately learn the behavior or playing style of a user, which can help for purposes such as player skill determination, as may be useful for matchmaking or difficulty setting, as well as training or recommendations that may be presented during a game. Learning how a player plays a game can also help to better understand which regions are likely to be indicative of certain events based on that player's style, as well as relative weightings to be given to those regions.

Inference and Training Logic

FIG. **7**A 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. **7**A and/or **7**B.

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.

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.

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.

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.

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.

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.

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. **7***a* 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. **7***a* 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").

FIG. **7***b* 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. **7***b* 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. **7***b* 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. **7***b*, 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**.

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

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**.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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. **7**A and/or **7**B. 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

Computer Systems

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, Calif., 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.

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.

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**.

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.

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.

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**.

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**.

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.

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.

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. **7**A and/or **7**B. 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

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.

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 1° C. 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.

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.

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").

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. **7***a* and/or **7***b*. 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

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.

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**.

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).

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.

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.

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.

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.

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**.

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. **7**A and/or **7**B. 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 FIG. **7**A or **7**B. 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

FIG. **12** is a block diagram of a processor **1200** having one or more processor cores **1202**A-**1202**N, 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 **1202**N represented by dashed lined boxes. In at least one embodiment, each of processor cores **1202**A-**1202**N includes one or more internal cache units **1204**A-**1204**N. In at least one embodiment, each processor core also has access to one or more shared cached units **1206**.

In at least one embodiment, internal cache units **1204**A-**1204**N and shared cache units **1206** represent a cache memory hierarchy within processor **1200**. In at least one embodiment, cache memory units **1204**A-**1204**N 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 **1204**A-**1204**N.

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).

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

tionally include a power control unit (PCU), which includes logic and components to regulate one or more power states of processor cores **1202**A-**1202**N and graphics processor **1208**.

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**.

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**.

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 **1202**A-**1202**N and graphics processor **1208** use embedded memory modules **1218** as a shared Last Level Cache.

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

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. **7***a* and/or **7***b*. 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) **1202**A-**1202**N, 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 FIG. **7**A or **7**B. 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

Virtualized Computing Platform

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.

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**.

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 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.

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.

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.

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.

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.

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**.

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.

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.

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).

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.

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.

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.

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.

In at least one embodiment, system **1400** (e.g., training system **1304** and/or deployment system **1306**) may 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.

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(ses), wireless data protocols (Wi-Fi), wired data protocols (e.g., Ethernet), etc.

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**.

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.

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. **15**B. 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.

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.

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 Mill machine, there may be a first deployment pipeline **1410**, and where image enhancement is desired from output of an Mill machine, there may be a second deployment pipeline **1410**.

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.

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**.

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. **12***cc*) 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.

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.

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.

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**.

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.

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.

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.

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.

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.).

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**.

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**.

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 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**.

FIG. **15**A 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**.

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**).

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.

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.

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**).

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**.

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.

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.

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.

FIG. **15**B 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 **1536**B in FIG. **15**B, 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.

Such components can be used to analyze specific regions of content in order to determine an occurrence of an event of interest without having to analyze all such regions. These events can be used for various purposes, such as to generate highlight sequences.

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.

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.

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."

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

What is claimed is:

1. A computer-implemented method, comprising:

analyzing, by a neural network, a first region of one or more frames of video content associated with a first element, wherein the first region is associated with a first level of a region hierarchy comprising multiple levels, each level defining one or more regions of the video content, the neural network trained to recognize objects using feature data extracted from video frames;

in response to determining that the first element in the first region has a first state associated with a type of event, analyzing, using the neural network, at least one second region of the one or more frames of video content, the second region associated with a second element and a second level of the region hierarchy;

wherein the analyzing is triggered by a determination that the second level is lower in the region hierarchy than the first level;

wherein at least one region associated with a lower level of the region hierarchy is analyzed in response to determining that an element in at least one region associated with a higher level of the region hierarchy is in a specific state associated with the type of event;

determining, using the neural network, that the second element has at least one second state associated with the type of event;

in response to determining that the first element is in the first state and the second element is in the least one second state associated with the type of event, identifying that the type of event occurred; and

providing a portion of the video content, representative of the type of event, for display on a client device.

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

determining the first region and the at least one second region using at least one rule for the type of event.

3. The computer-implemented method of claim **2**, further comprising:

analyzing one or more objects represented in the video content to determine the type of event; and

determining the at least one rule corresponding to the type of event.

4. The computer-implemented method of claim **1**, further comprising:

analyzing the first region for each video frame as part of a first pass; and

analyzing the at least one second region as part of a second pass performed on a video frame only when the first element of the first region has the first state associated with the type of event.

5. The computer-implemented method of claim **1**, wherein the at least one second region is capable of being located at one or more secondary levels below a primary level containing the first region, and further comprising:

evaluating individual second regions only when a parent region has a state associated with the type of event.

6. The computer-implemented method of claim **1**, wherein the first element is one of a plurality of first elements having at least one state associated with the type of event, and wherein each of the plurality of first elements is analyzed on a first pass for each individual video frame.

7. The computer-implemented method of claim **1**, wherein the portion of the video content includes at least one of a highlight sequence, a video montage, a training video, a game summary, player statistics, or a player skill profile.

8. The computer-implemented method of claim **1**, wherein the first element and the second element include at least one of an icon, a graphical element, text, or audio content, and wherein the first state associated with a type of event is capable of being determined relative to a prior state of the first element.

9. The computer-implemented method of claim **1**, further comprising:

determining whether the portion of the video content should be provided for display on the client device based on at least one highlight selection criterion.

10. The computer-implemented method of claim **1**, wherein the first region and the at least one second region correspond to elements of a user interface or a heads up display (HUD).

11. A highlight generation system, comprising:

at least one processor; and

memory including instructions that, when executed by the at least one processor, cause the highlight generation system to:

analyze, by a neural network, a first region of one or more frames of video content associated with a first element, wherein the first region is associated with a first level of a region hierarchy comprising multiple levels, each level defining one or more regions of the video content, the neural network trained to recognize objects using feature data extracted from video frames;

in response to determining that the first element in the first region has a first state associated with a type of event analyze, use the neural network, at least one second region of the one or more frames of video

content, the second region associated with a second element and a second level of the region hierarch;

wherein the analyzing is triggered based on a determination that the second level is lower in the region hierarchy than the first level; and

wherein at least one region associated with a lower level of the region hierarchy is analyzed in response to determining that an element in at least one region associated with a higher level of the region hierarchy is in a specific state associated with the type of event;

determine, using the neural network, that the second element has at least one second state associated with the type of event;

in response to determining that the first element is in the first state and the second element is in the at least one second state associated with the type of event, identify that the type of event occurred; and

provide a portion of the video content, representative of the type of event, for display on a client device.

12. The highlight generation system of claim **11**, wherein the instructions when executed further cause the highlight generation system to:

determine the first region and the at least one second region using at least one rule for the type of event associated with a content type of the video content.

13. The highlight generation system of claim **11**, wherein the instructions when executed further cause the highlight generation system to:

analyze the first region for each video frame as part of a first pass; and

analyze the at least one second region as part of a second pass performed on a video frame only when the first element of the first region has the first state associated with the type of event.

14. The highlight generation system of claim **11**, wherein the instructions when executed further cause the highlight generation system to:

evaluate individual second regions only when a parent region has a state associated with the type of event.

15. The highlight generation system of claim **11**, wherein the video content corresponds to a video game, virtual reality (VR) experience, augmented reality (AR) experience, mixed reality (MR) experience, animation, or captured performance.

16. A processor, comprising:

one or more processing units to:

determine, using a neural network, a portion of video content representative of an occurrence of a type of event;

analyze one or more frames of the video content using a set of rules associated with the type of event to determine that a first element associated with a respective first region associated with a first level in a region hierarchy has a first state indicative of the occurrence, and that at least one second element associated with at least one respective second region associated with a second level in the region hierarchy has at least a second state indicative of the occurrence, wherein analyzing the second region is triggered by the first element associated with the first region having the first state indicative of the occurrence and by a determination that the second level is lower in the region hierarchy than the first level; and

provide at least a portion of the one or more frames of the video content, representative of the type of event, for display on a client device,

wherein the region hierarchy is associated with the type of event; and

wherein at least one region associated with a lower level of the region hierarchy is analyzed if it is determined that an element in at least one region associated with a higher level of the region hierarchy is in a state associated with the type of event.

17. The processor of claim **16**, wherein at least one processing unit of the one or more processing units is further configured to:

analyze the first region for individual frames of the video content as part of a first pass; and

analyze the at least one second region as part of a second pass performed on a video frame only when the first element of the first region has the first state associated with the type of event.

18. The processor of claim **16**, wherein the portion of the video content includes at least one of a highlight sequence, a video montage, a training video, a game summary, player statistics, or a player skill profile.

19. The processor of claim **16**, wherein the first element and the at least one second element include at least one of an icon, a graphical element, text, or audio content, and wherein the first state associated with the event of interest is capable of being determined relative to a prior state of the first element.

20. The processor of claim **16**, wherein the first element is one of a plurality of first elements having at least one state associated with the type of event, and wherein each of the plurality of first elements is analyzed on a first pass for individual frames of the video content.

* * * * *