US 20200302250A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2020/0302250 A1

Chu et al. (43) Pub. Date: Sep. 24, 2020

(54) ITERATIVE SPATIAL GRAPH GENERATION

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

(72) Inventors: **Hang Chu**, Toronto (CA); **Daiqing Li**, Toronto (CA); **David Jesus Acuna Marrero**, Toronto (CA); **Amlan Kar**, Toronto (CA); **Maria Shugrina**, Toronto (CA); **Ming-Yu Liu**, San Jose, CA (US); **Antonio Torralba Barriuso**, Somerville, MA (US); **Sanja Fidler**, Toronto (CA)

(21) Appl. No.: **16/825,199**

(22) Filed: **Mar. 20, 2020**

Related U.S. Application Data

(60) Provisional application No. 62/822,754, filed on Mar. 22, 2019.

Publication Classification

(51) **Int. Cl.**

G06K 9/62	(2006.01)
G06N 3/08	(2006.01)
G06N 3/04	(2006.01)
G06N 5/04	(2006.01)
G06N 20/10	(2006.01)
G06N 20/20	(2006.01)

(52) **U.S. Cl.**

CPC ***G06K 9/6296*** (2013.01); ***G06N 3/084*** (2013.01); ***G06N 20/20*** (2019.01); ***G06N 5/04*** (2013.01); ***G06N 20/10*** (2019.01); ***G06N 3/0454*** (2013.01)

(57) **ABSTRACT**

A generative model can be used for generation of spatial layouts and graphs. Such a model can progressively grow these layouts and graphs based on local statistics, where nodes can represent spatial control points of the layout, and edges can represent segments or paths between nodes, such as may correspond to road segments. A generative model can utilize an encoder-decoder architecture where the encoder is a recurrent neural network (RNN) that encodes local incoming paths into a node and the decoder is another RNN that generates outgoing nodes and edges connecting an existing node to the newly generated nodes. Generation is done iteratively, and can finish once all nodes are visited or another end condition is satisfied. Such a model can generate layouts by additionally conditioning on a set of attributes, giving control to a user in generating the layout.

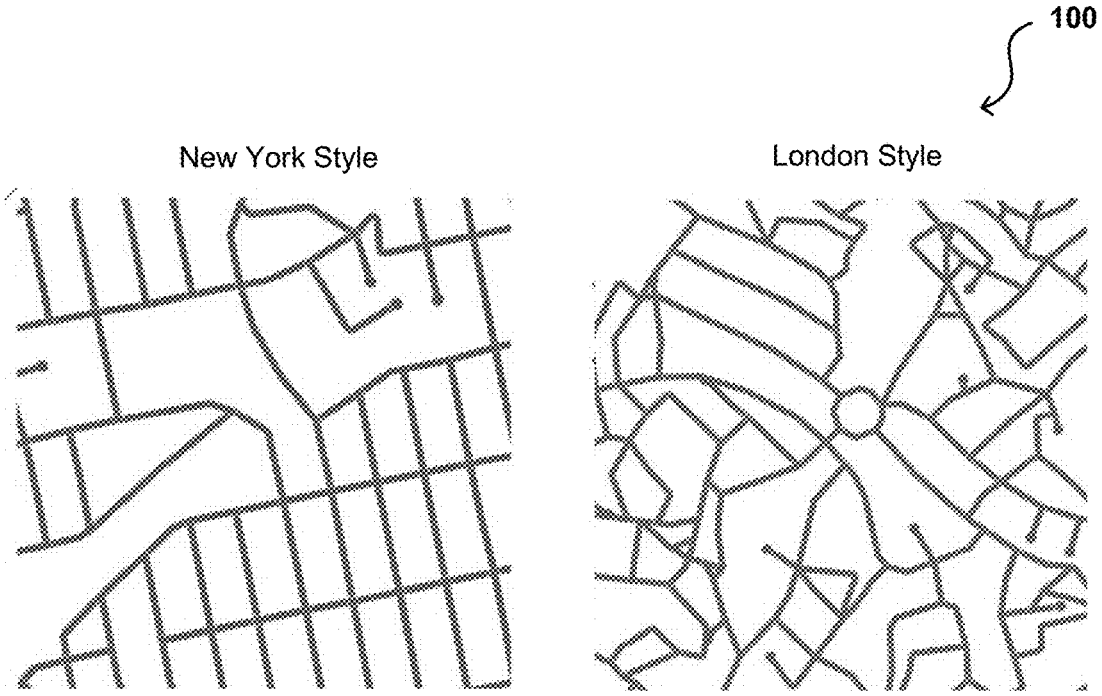
100
New York Style
London Style

FIG. 1A

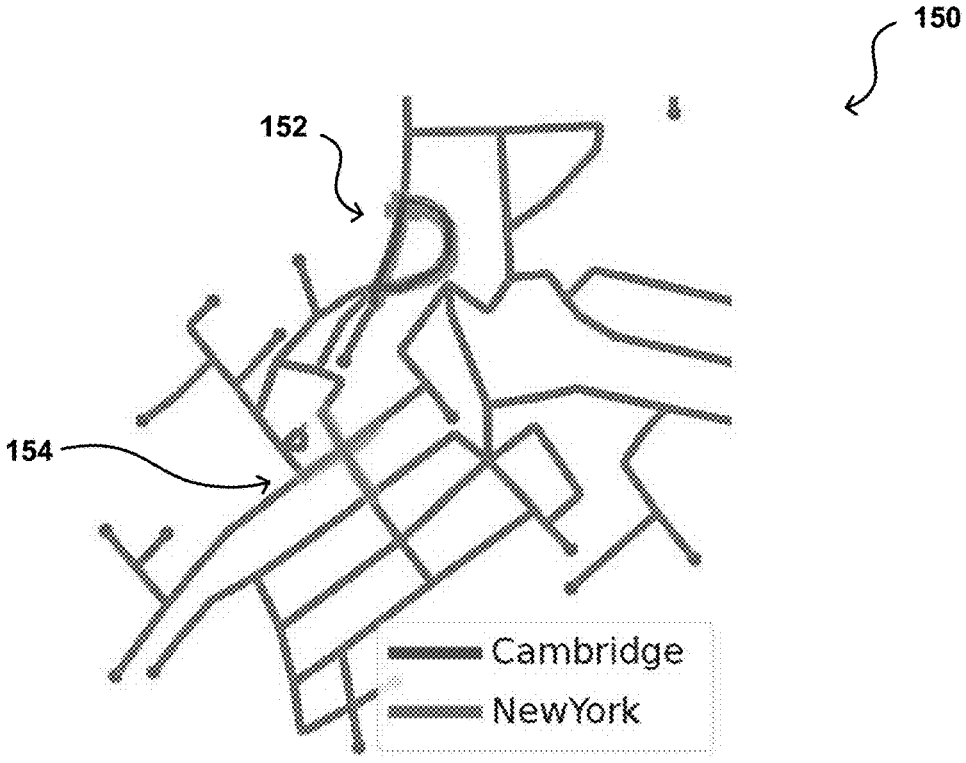
150
152
154
Cambridge
NewYork

FIG. 1B

FIG. 2

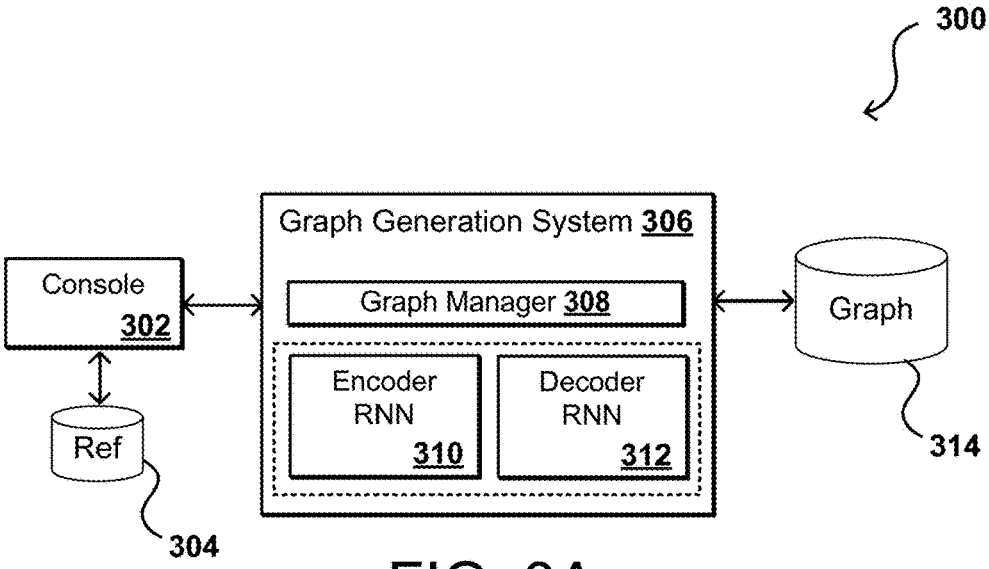
300
Graph Generation System 306
Console 302
Graph Manager 308
Graph
Encoder RNN 310
Decoder RNN 312
Ref
304
314

FIG. 3A

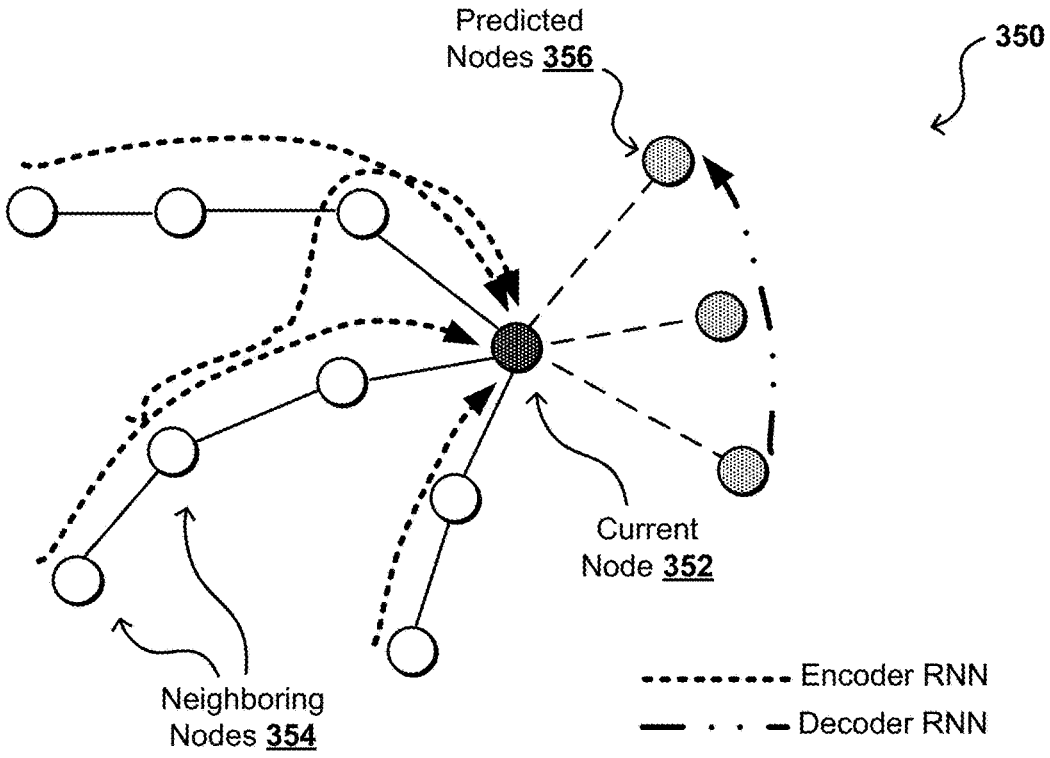
Predicted Nodes 356
350
Current Node 352
Neighboring Nodes 354
Encoder RNN
Decoder RNN

FIG. 3B

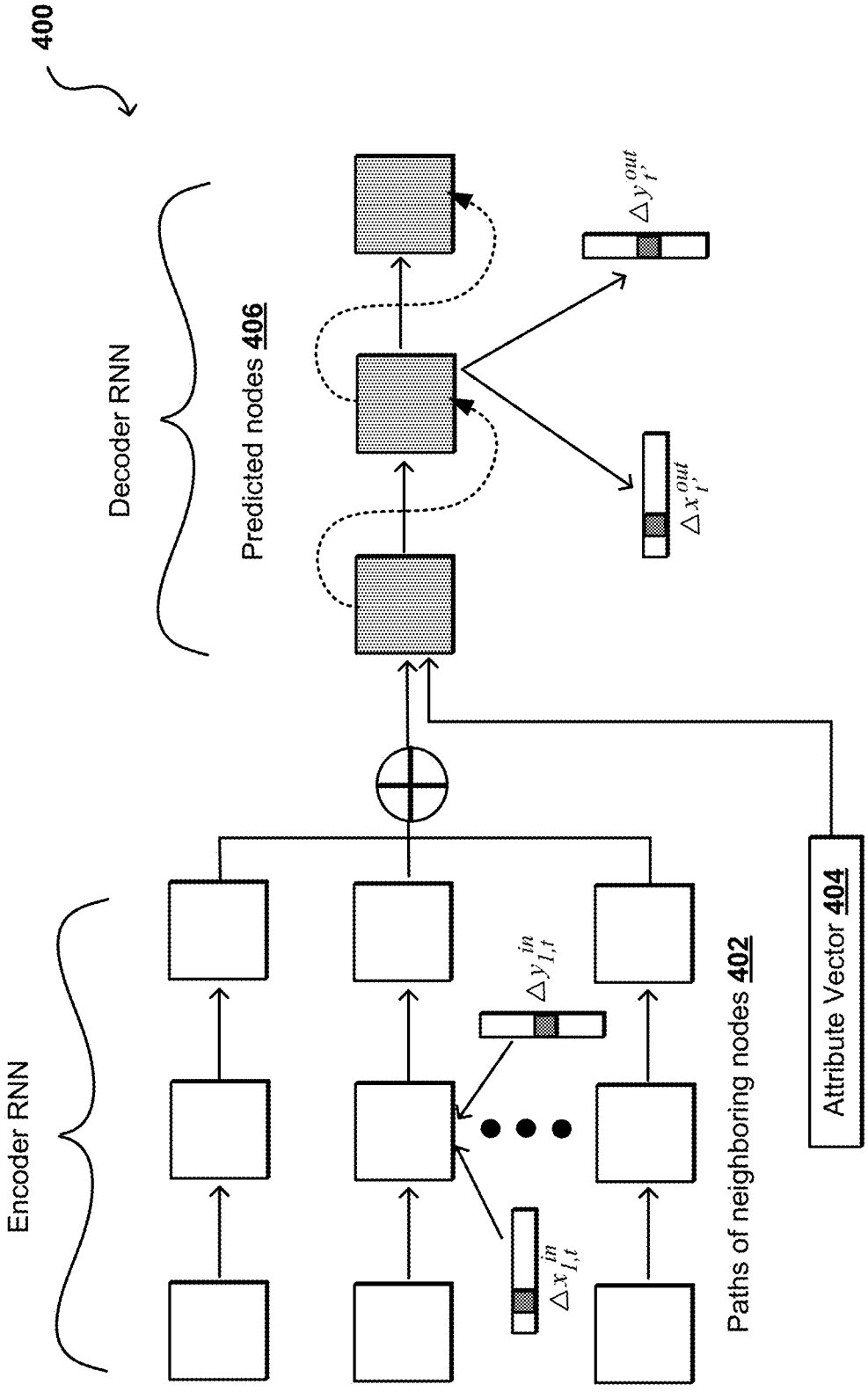
400
Encoder RNN
Decoder RNN
Predicted nodes 406
Paths of neighboring nodes 402
Attribute Vector 404
$\Delta x_{1,t}^{in}$
$\Delta y_{1,t}^{in}$
$\Delta x_{t'}^{out}$
$\Delta y_{t'}^{out}$

FIG. 4

FIG. 5

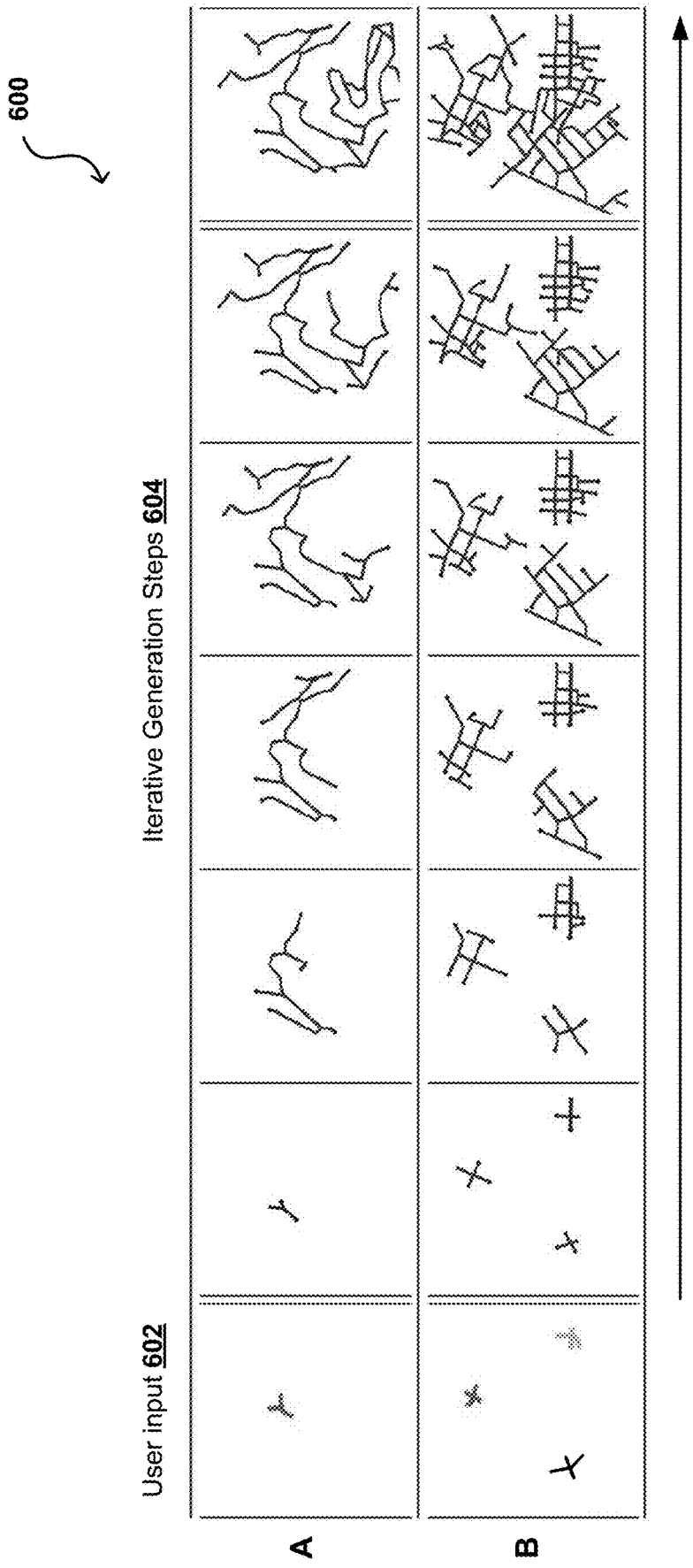

FIG. 6

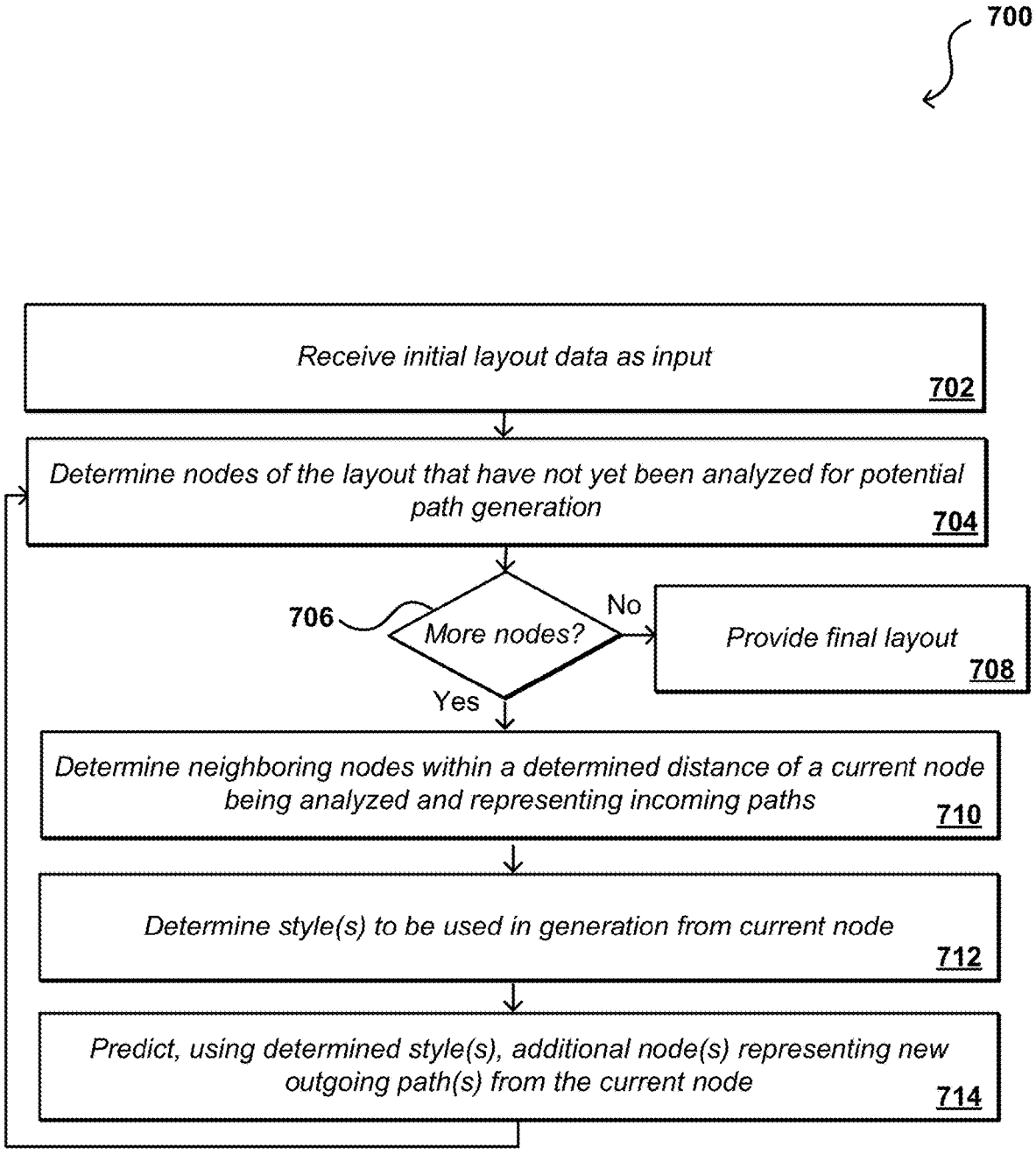
700
Receive initial layout data as input
702
Determine nodes of the layout that have not yet been analyzed for potential path generation
704
706
More nodes?
No
Provide final layout
708
Yes
Determine neighboring nodes within a determined distance of a current node being analyzed and representing incoming paths
710
Determine style(s) to be used in generation from current node
712
Predict, using determined style(s), additional node(s) representing new outgoing path(s) from the current node
714

FIG. 7

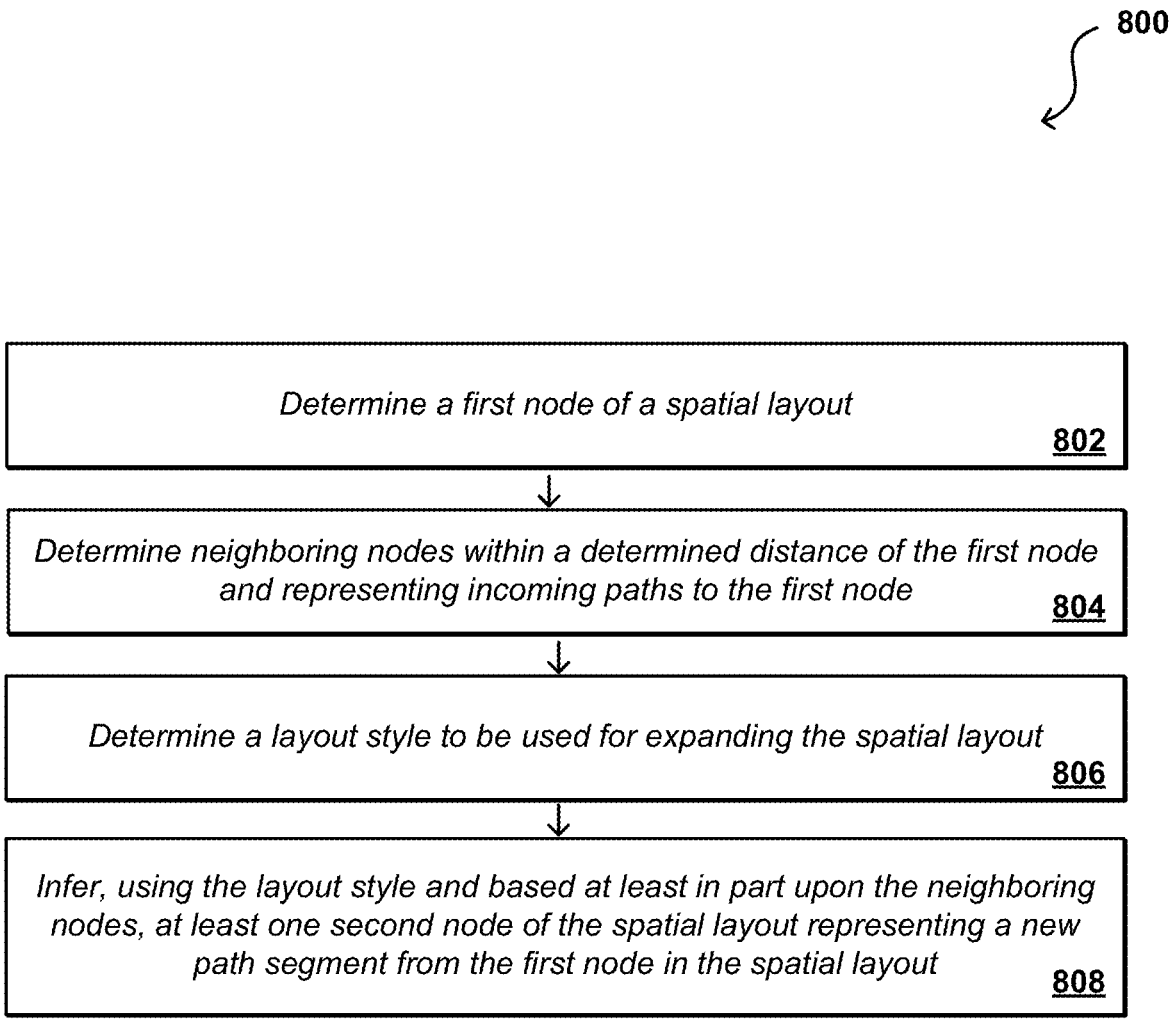
800
Determine a first node of a spatial layout
802
Determine neighboring nodes within a determined distance of the first node and representing incoming paths to the first node
804
Determine a layout style to be used for expanding the spatial layout
806
Infer, using the layout style and based at least in part upon the neighboring nodes, at least one second node of the spatial layout representing a new path segment from the first node in the spatial layout
808

FIG. 8

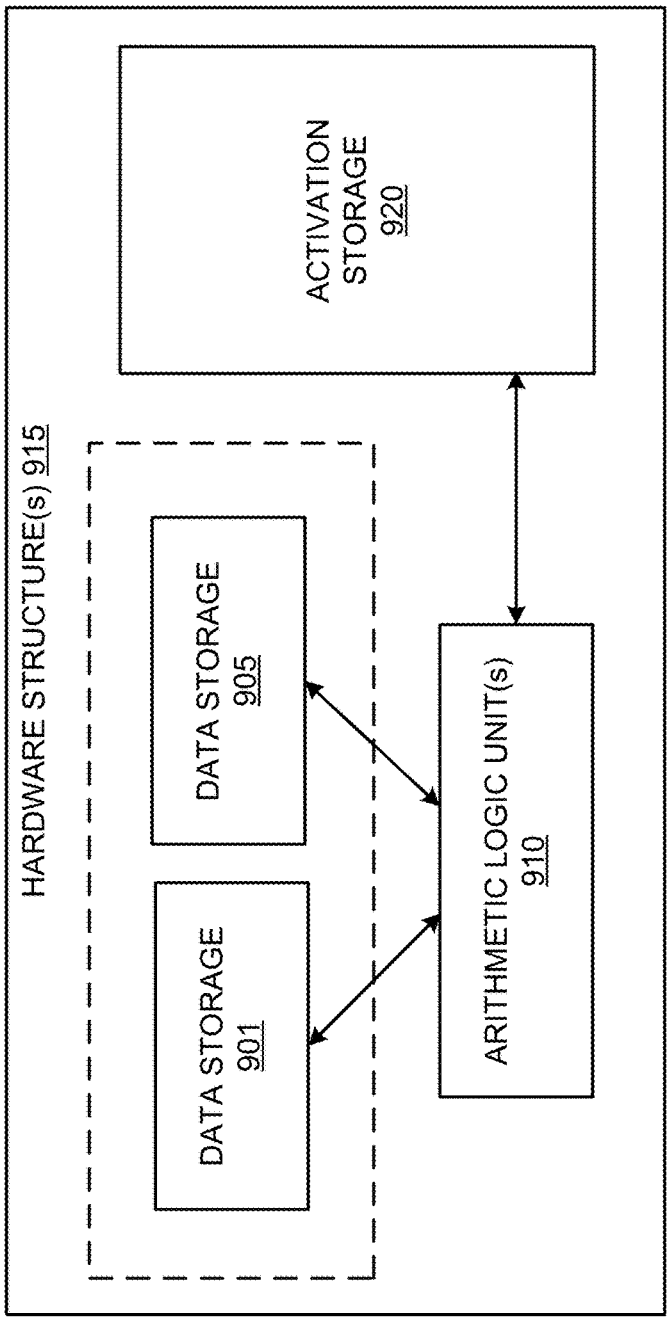
HARDWARE STRUCTURE(s) 915
ACTIVATION STORAGE 920
DATA STORAGE 905
DATA STORAGE 901
ARITHMETIC LOGIC UNIT(s) 910

FIG. 9A

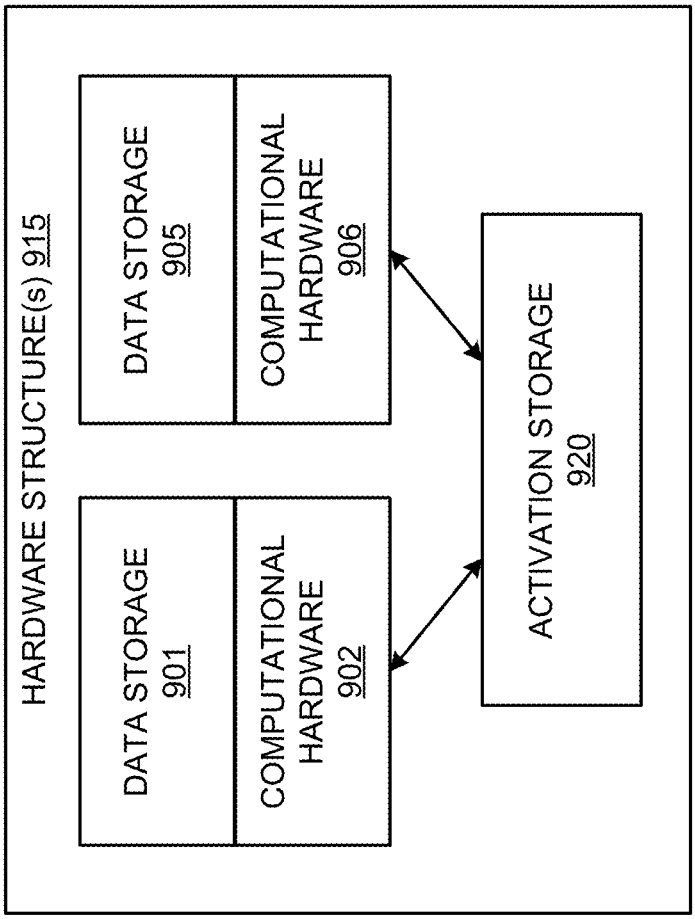
HARDWARE STRUCTURE(s) 915
DATA STORAGE 901
COMPUTATIONAL HARDWARE 902
DATA STORAGE 905
COMPUTATIONAL HARDWARE 906
ACTIVATION STORAGE 920

FIG. 9B

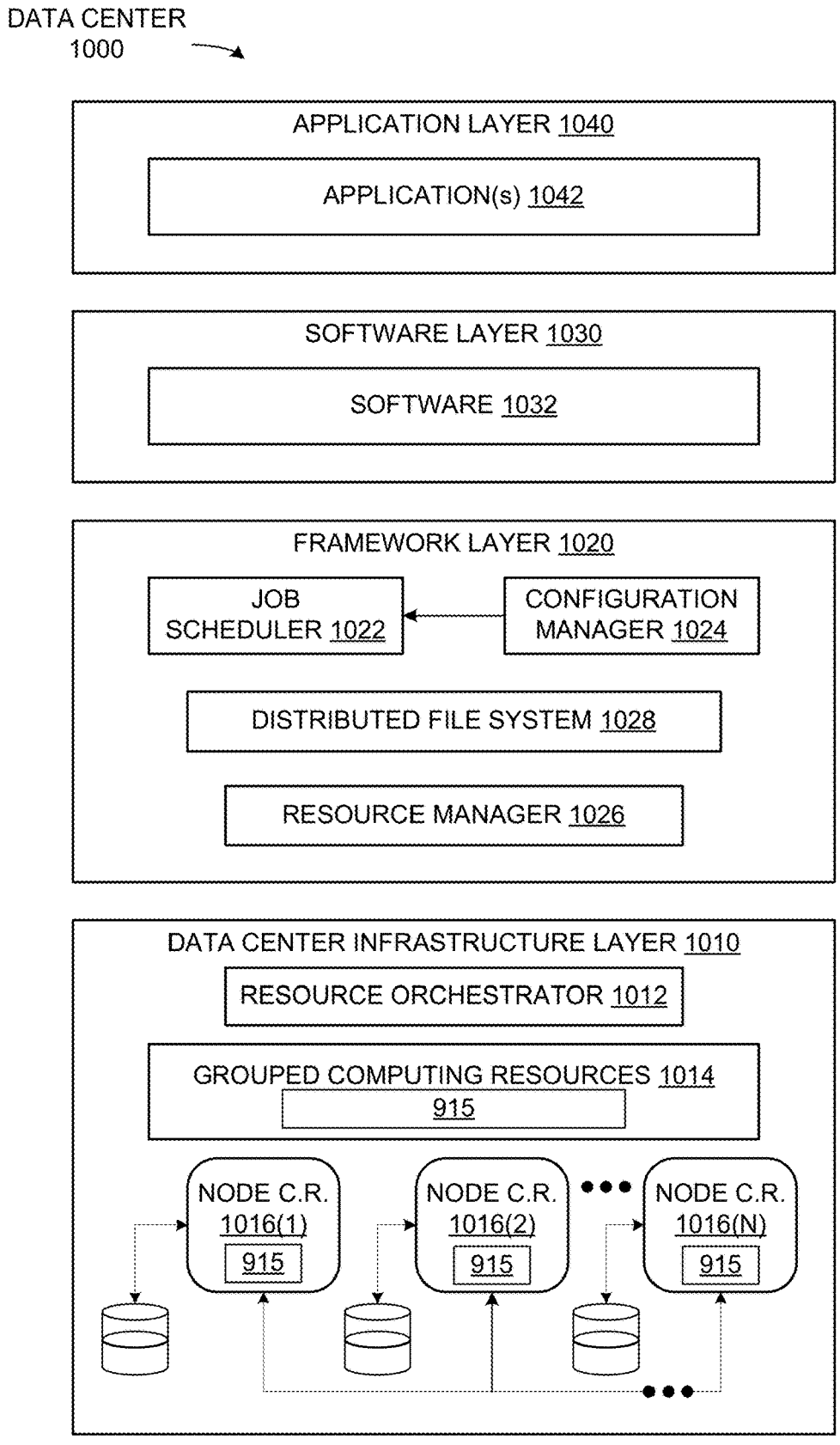
DATA CENTER
1000
APPLICATION LAYER 1040
APPLICATION(s) 1042
SOFTWARE LAYER 1030
SOFTWARE 1032
FRAMEWORK LAYER 1020
JOB SCHEDULER 1022
CONFIGURATION MANAGER 1024
DISTRIBUTED FILE SYSTEM 1028
RESOURCE MANAGER 1026
DATA CENTER INFRASTRUCTURE LAYER 1010
RESOURCE ORCHESTRATOR 1012
GROUPED COMPUTING RESOURCES 1014
915
NODE C.R. 1016(1)
915
NODE C.R. 1016(2)
915
NODE C.R. 1016(N)
915

FIG. 10

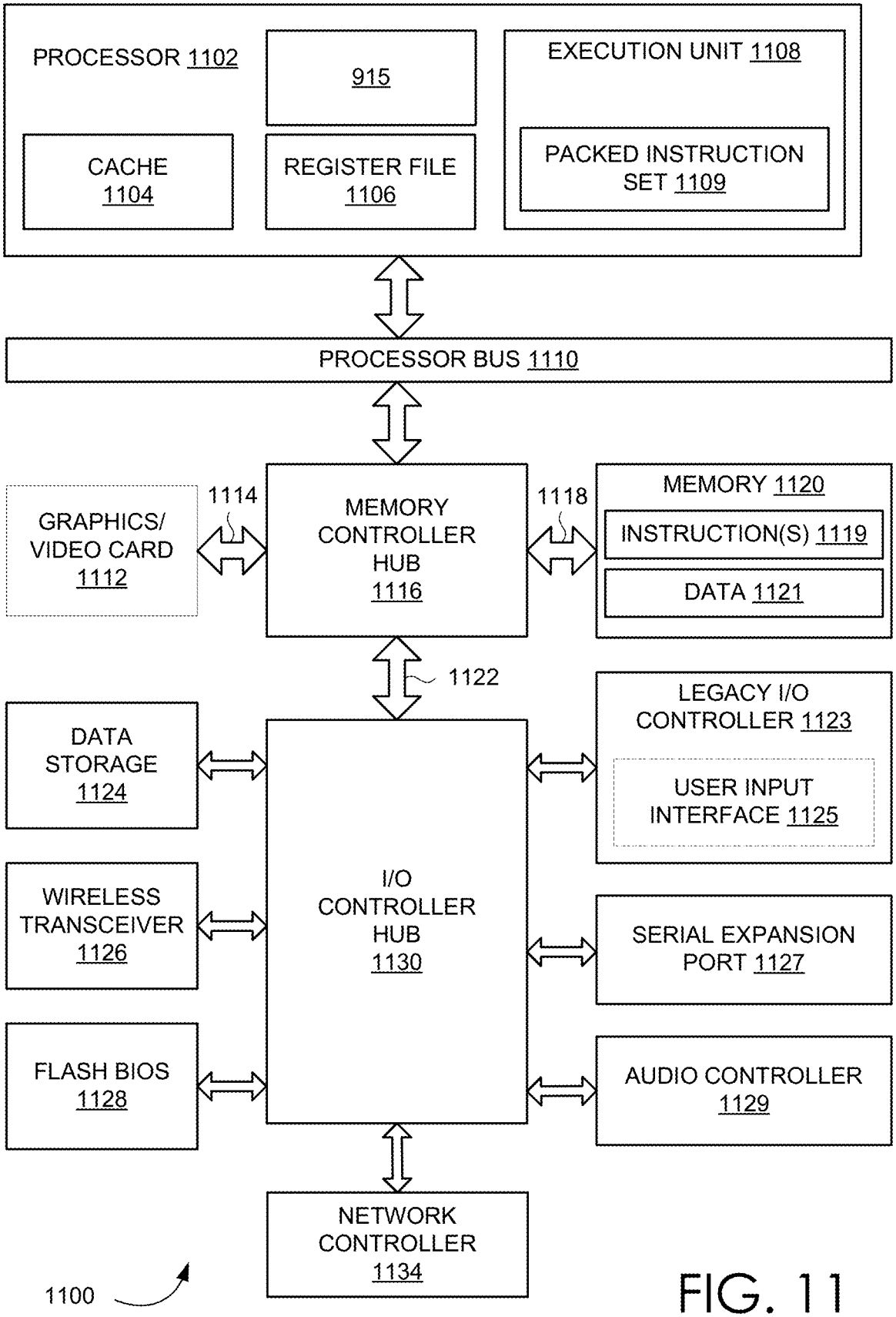
PROCESSOR 1102
915
EXECUTION UNIT 1108
CACHE 1104
REGISTER FILE 1106
PACKED INSTRUCTION SET 1109
PROCESSOR BUS 1110
GRAPHICS/ VIDEO CARD 1112
1114
MEMORY CONTROLLER HUB 1116
1118
MEMORY 1120
INSTRUCTION(S) 1119
DATA 1121
1122
DATA STORAGE 1124
LEGACY I/O CONTROLLER 1123
USER INPUT INTERFACE 1125
WIRELESS TRANSCEIVER 1126
I/O CONTROLLER HUB 1130
SERIAL EXPANSION PORT 1127
FLASH BIOS 1128
AUDIO CONTROLLER 1129
NETWORK CONTROLLER 1134
1100

FIG. 11

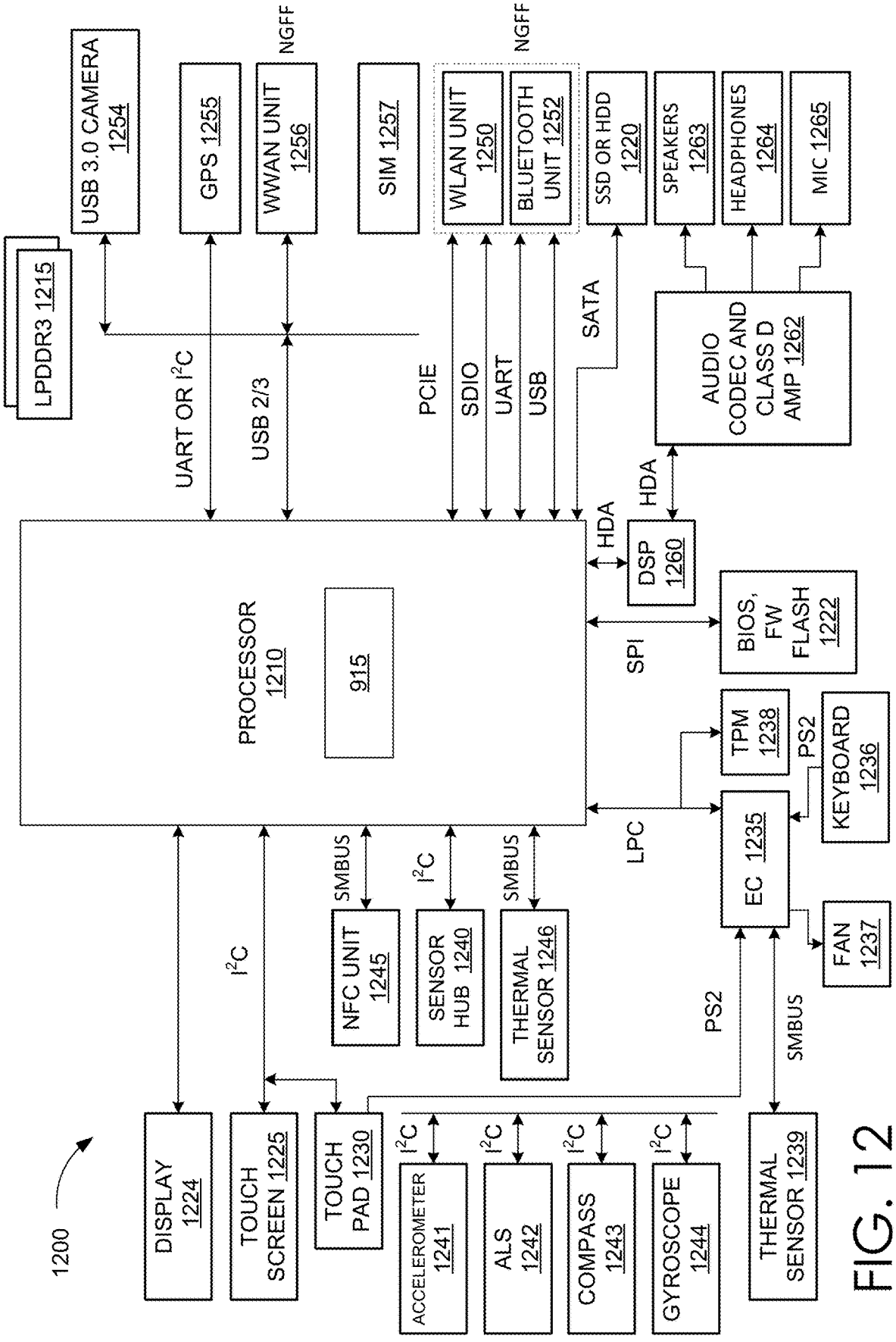
1200
DISPLAY 1224
TOUCH SCREEN 1225
TOUCH PAD 1230
ACCELEROMETER 1241
ALS 1242
COMPASS 1243
GYROSCOPE 1244
THERMAL SENSOR 1239
I2C
NFC UNIT 1245
SENSOR HUB 1240
THERMAL SENSOR 1246
SMBUS
I2C
SMBUS
PROCESSOR 1210
915
LPC
EC 1235
PS2
SMBUS
TPM 1238
PS2
KEYBOARD 1236
FAN 1237
SPI
BIOS, FW FLASH 1222
HDA
DSP 1260
HDA
AUDIO CODEC AND CLASS D AMP 1262
SATA
UART OR I2C
USB 2/3
PCIE
SDIO
UART
USB
LPDDR3 1215
USB 3.0 CAMERA 1254
GPS 1255
NGFF
WWAN UNIT 1256
SIM 1257
NGFF
WLAN UNIT 1250
BLUETOOTH UNIT 1252
SSD OR HDD 1220
SPEAKERS 1263
HEADPHONES 1264
MIC 1265

FIG. 12

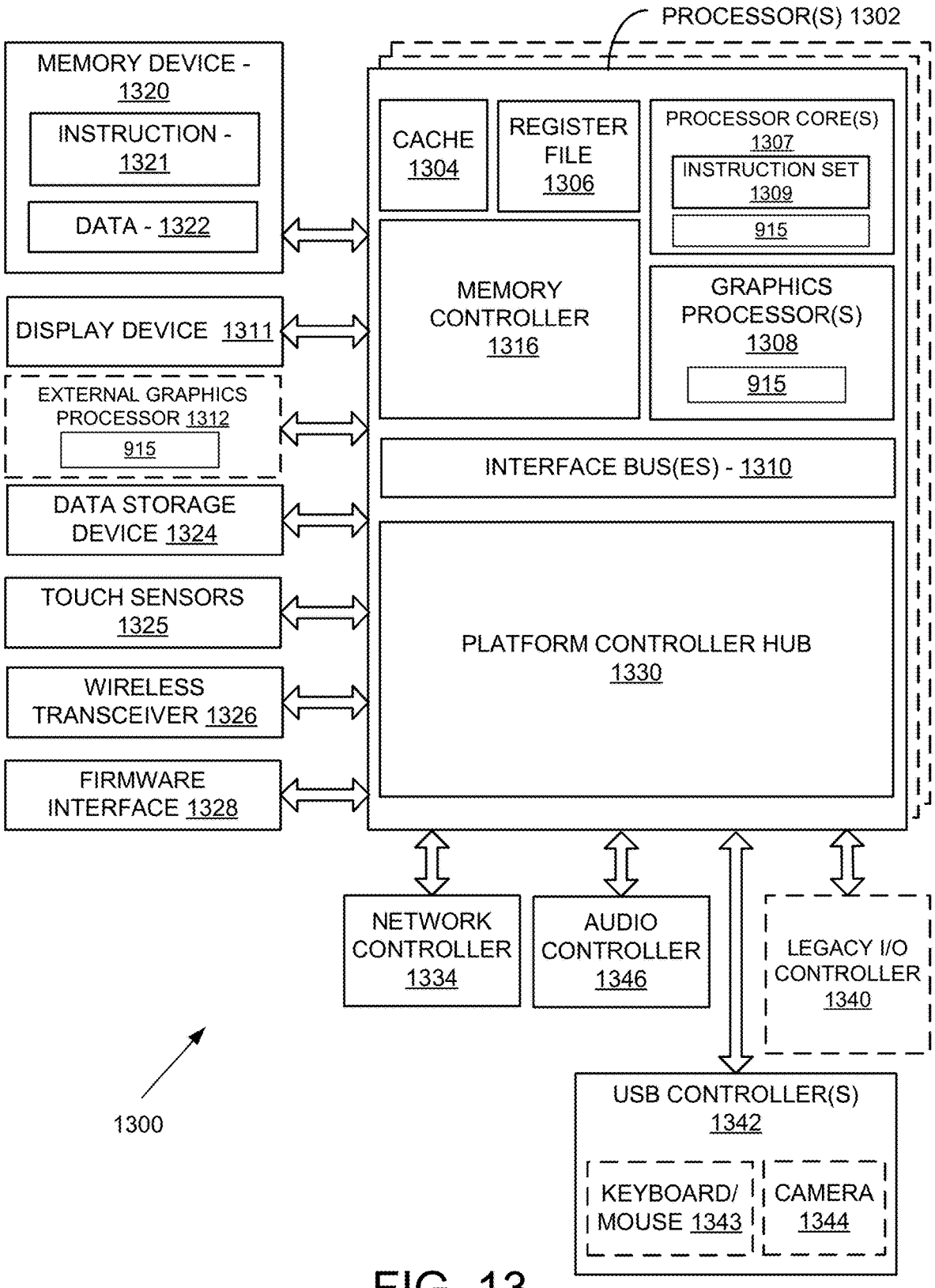
PROCESSOR(S) 1302
MEMORY DEVICE - 1320
INSTRUCTION - 1321
DATA - 1322
CACHE 1304
REGISTER FILE 1306
PROCESSOR CORE(S) 1307
INSTRUCTION SET 1309
915
MEMORY CONTROLLER 1316
GRAPHICS PROCESSOR(S) 1308
915
DISPLAY DEVICE 1311
EXTERNAL GRAPHICS PROCESSOR 1312
915
INTERFACE BUS(ES) - 1310
DATA STORAGE DEVICE 1324
TOUCH SENSORS 1325
PLATFORM CONTROLLER HUB 1330
WIRELESS TRANSCEIVER 1326
FIRMWARE INTERFACE 1328
NETWORK CONTROLLER 1334
AUDIO CONTROLLER 1346
LEGACY I/O CONTROLLER 1340
USB CONTROLLER(S) 1342
KEYBOARD/ MOUSE 1343
CAMERA 1344
1300

FIG. 13

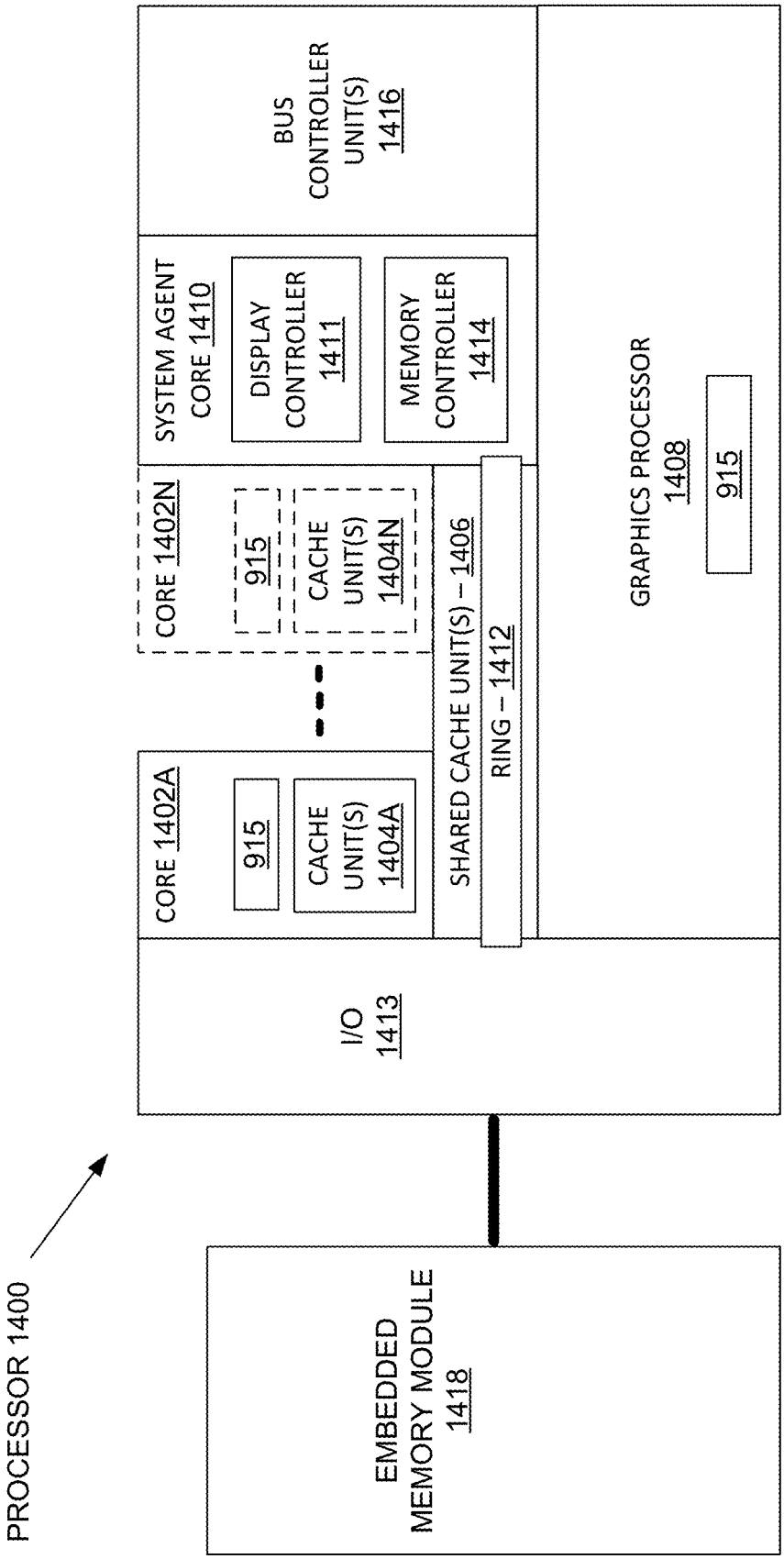
PROCESSOR 1400
EMBEDDED MEMORY MODULE 1418
I/O 1413
CORE 1402A
915
CACHE UNIT(S) 1404A
CORE 1402N
915
CACHE UNIT(S) 1404N
SHARED CACHE UNIT(S) – 1406
RING – 1412
SYSTEM AGENT CORE 1410
DISPLAY CONTROLLER 1411
MEMORY CONTROLLER 1414
BUS CONTROLLER UNIT(S) 1416
GRAPHICS PROCESSOR 1408
915

FIG. 14

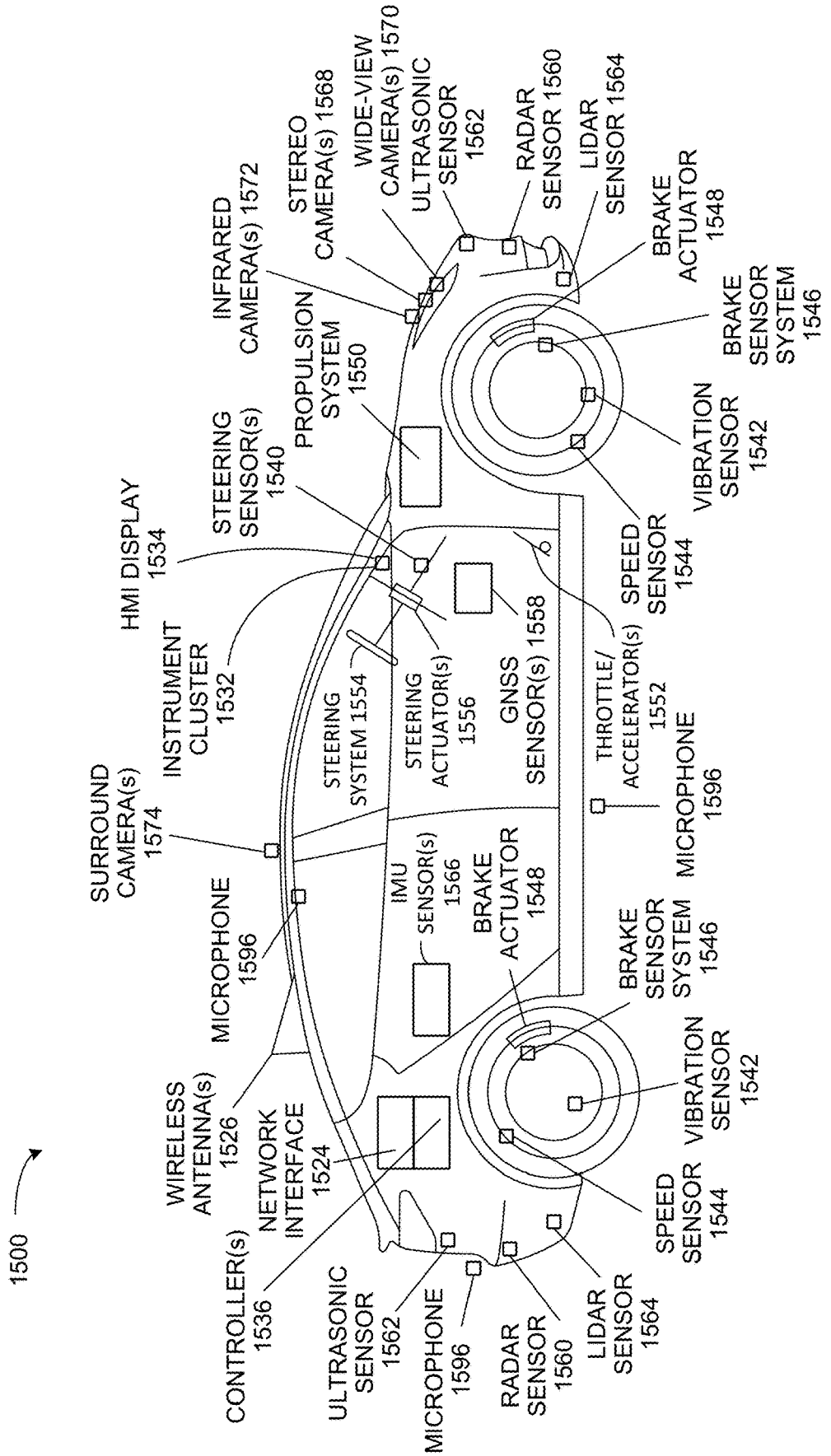
1500
WIRELESS ANTENNA(S) 1526
NETWORK INTERFACE 1524
CONTROLLER(S) 1536
ULTRASONIC SENSOR 1562
MICROPHONE 1596
RADAR SENSOR 1560
LIDAR SENSOR 1564
SPEED SENSOR 1544
VIBRATION SENSOR 1542
BRAKE SENSOR SYSTEM 1546
BRAKE ACTUATOR 1548
IMU SENSOR(S) 1566
MICROPHONE 1596
SURROUND CAMERA(S) 1574
INSTRUMENT CLUSTER 1532
HMI DISPLAY 1534
STEERING SENSOR(S) 1540
INFRARED CAMERA(S) 1572
STEREO CAMERA(S) 1568
WIDE-VIEW CAMERA(S) 1570
ULTRASONIC SENSOR 1562
RADAR SENSOR 1560
LIDAR SENSOR 1564
BRAKE ACTUATOR 1548
BRAKE SENSOR SYSTEM 1546
VIBRATION SENSOR 1542
SPEED SENSOR 1544
PROPULSION SYSTEM 1550
STEERING SYSTEM 1554
STEERING ACTUATOR(S) 1556
GNSS SENSOR(S) 1558
THROTTLE/ ACCELERATOR(S) 1552
MICROPHONE 1596

FIG. 15A

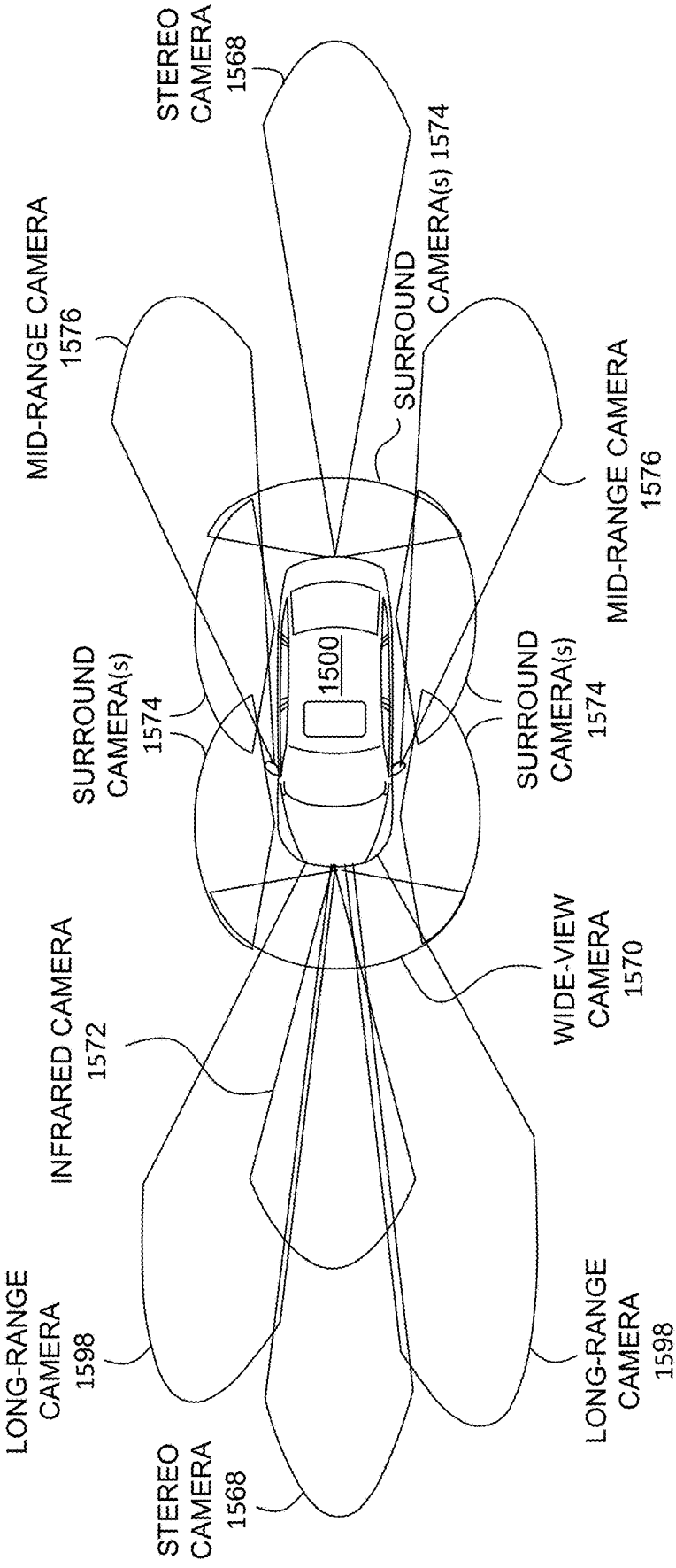
STEREO CAMERA 1568
MID-RANGE CAMERA 1576
SURROUND CAMERA(s) 1574
1500
SURROUND CAMERA(s) 1574
MID-RANGE CAMERA 1576
INFRARED CAMERA 1572
WIDE-VIEW CAMERA 1570
LONG-RANGE CAMERA 1598
STEREO CAMERA 1568
LONG-RANGE CAMERA 1598

FIG. 15B

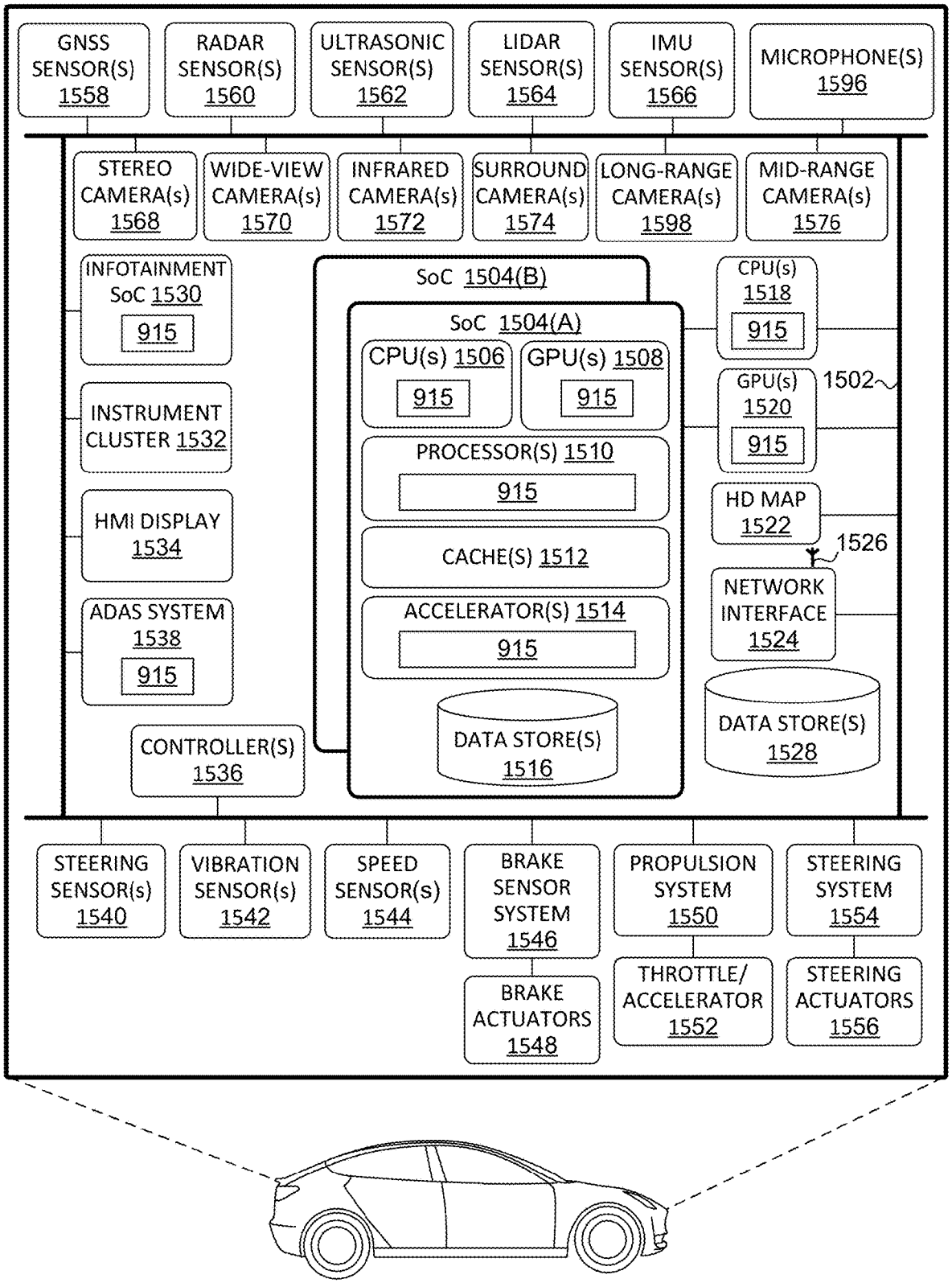
GNSS SENSOR(S) 1558
RADAR SENSOR(S) 1560
ULTRASONIC SENSOR(S) 1562
LIDAR SENSOR(S) 1564
IMU SENSOR(S) 1566
MICROPHONE(S) 1596
STEREO CAMERA(s) 1568
WIDE-VIEW CAMERA(s) 1570
INFRARED CAMERA(s) 1572
SURROUND CAMERA(s) 1574
LONG-RANGE CAMERA(s) 1598
MID-RANGE CAMERA(s) 1576
INFOTAINMENT SoC 1530
915
INSTRUMENT CLUSTER 1532
HMI DISPLAY 1534
ADAS SYSTEM 1538
915
CONTROLLER(S) 1536
SoC 1504(B)
SoC 1504(A)
CPU(s) 1506
915
GPU(s) 1508
915
PROCESSOR(S) 1510
915
CACHE(S) 1512
ACCELERATOR(S) 1514
915
DATA STORE(S) 1516
CPU(s) 1518
915
GPU(s) 1520
915
1502
HD MAP 1522
1526
NETWORK INTERFACE 1524
DATA STORE(S) 1528
STEERING SENSOR(s) 1540
VIBRATION SENSOR(s) 1542
SPEED SENSOR(s) 1544
BRAKE SENSOR SYSTEM 1546
BRAKE ACTUATORS 1548
PROPULSION SYSTEM 1550
THROTTLE/ ACCELERATOR 1552
STEERING SYSTEM 1554
STEERING ACTUATORS 1556

FIG. 15C

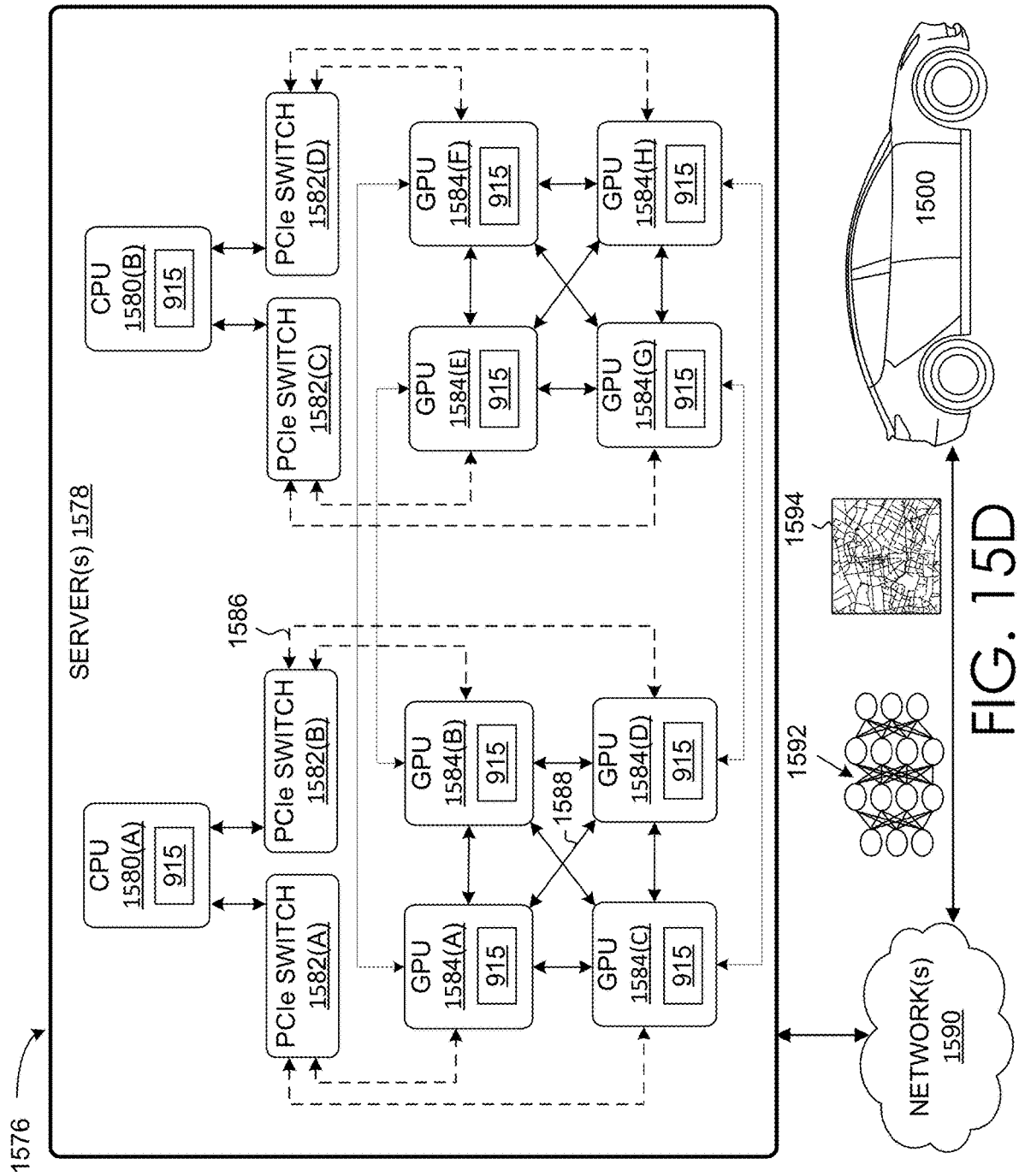
1576
SERVER(s) 1578
CPU 1580(A) 915
CPU 1580(B) 915
PCIe SWITCH 1582(A)
PCIe SWITCH 1582(B)
PCIe SWITCH 1582(C)
PCIe SWITCH 1582(D)
1586
1588
GPU 1584(A) 915
GPU 1584(B) 915
GPU 1584(C) 915
GPU 1584(D) 915
GPU 1584(E) 915
GPU 1584(F) 915
GPU 1584(G) 915
GPU 1584(H) 915
1592
1594
1500
NETWORK(s) 1590

FIG. 15D

ITERATIVE SPATIAL GRAPH GENERATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Patent Application Ser. No. 62/822,754, filed Mar. 22, 2019, and entitled "Neural Turtle Graphics for Modeling City Road Layouts," which is hereby incorporated herein by reference in its entirety and for all purposes.

BACKGROUND

[0002] Layout modeling is an important problem in various fields. For example, extensive simulation of city layouts is required in urban planning to ensure that construction leads to effective traffic flow and connectivity. Layout modeling also finds demand in industries such as gaming and autonomous navigation, among others. Existing approaches to layout generation are largely based on procedural modeling with hand-designed features. Such approaches can be time consuming and inflexible, which limits their value and usefulness.

BRIEF DESCRIPTION OF THE DRAWINGS

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

[0004] FIGS. **1**A and **1**B illustrate spatial layouts that can be generated, according to at least one embodiment;

[0005] FIG. **2** illustrates road mappings that can be generated, according to at least one embodiment;

[0006] FIGS. **3**A and **3**B illustrate components of a layout generation system and related functionality, according to at least one embodiment;

[0007] FIG. **4** illustrates node prediction that can be performed, according to at least one embodiment;

[0008] FIG. **5** illustrates combinatorial layout generation, according to at least one embodiment;

[0009] FIG. **6** illustrates steps of an interactive layout generation process, according to at least one embodiment;

[0010] FIG. **7** illustrates a process for spatial layout generation, according to at least one embodiment;

[0011] FIG. **8** illustrates a process for expanding a spatial layout, according to at least one embodiment;

[0012] FIG. **9**A illustrates inference and/or training logic, according to at least one embodiment;

[0013] FIG. **9**B illustrates inference and/or training logic, according to at least one embodiment;

[0014] FIG. **10** illustrates an example data center system, according to at least one embodiment;

[0015] FIG. **11** illustrates a computer system, according to at least one embodiment;

[0016] FIG. **12** illustrates a computer system, according to at least one embodiment;

[0017] FIG. **13** illustrates at least portions of a graphics processor, according to one or more embodiments;

[0018] FIG. **14** illustrates at least portions of a graphics processor, according to one or more embodiments;

[0019] FIG. **15**A illustrates an example of an autonomous vehicle, according to at least one embodiment;

[0020] FIG. **15**B illustrates an example of camera locations and fields of view for the autonomous vehicle of FIG. **15**A, according to at least one embodiment;

[0021] FIG. **15**C illustrates an example system architecture for the autonomous vehicle of FIG. **15**A, according to at least one embodiment; and

[0022] FIG. **15**D illustrates a system for communication between cloud-based server(s) and the autonomous vehicle of FIG. **15**A, according to at least one embodiment.

DETAILED DESCRIPTION

[0023] As mentioned, layout modeling and spatial graphing play important roles in a variety of different industries and applications. For example, city road layout modeling is an important problem with applications in fields such as urban planning, where extensive simulation of city layouts is required to ensure that the final construction leads to effective traffic flow and connectivity. Although data-driven end-to-end learning paradigms have revolutionized various computer vision fields, existing approaches for city layout generation are still largely based on procedural modeling with hand-designed features. While these methods may provide valid road topologies with user specified attribute inputs, the attributes are all hand-engineered and inflexible to use. For example, if one wishes to generate a synthetic city that resembles the city of London, tedious manual tuning of the attributes is required in order to obtain plausible results. Moreover, these methods cannot trivially be used for applications such as aerial road parsing.

[0024] Accordingly, approaches in accordance with various embodiments provide for an iterative generation or expansion of spatial layouts. In at least one embodiment, generation can be performed using a generative model for spatial graphs, which in at least one embodiment can take the form of a neural vector graphics model. This can include an encoder-decoder neural network which operates on graphs locally for generating large spatial graphs. Such a model can generate spatial layouts as a graph of nodes connected by edges. In an application such as road modeling, the graph can represent a city road layout, the nodes can represent spatial control points along the roads, and the edges in the graphs can represent segments of roads or otherwise navigable paths. In at least one embodiment, a sequential, generative model can be used that is parameterized using a neural network. Such a model can iteratively generate a new node and an edge connecting to an existing node conditioned on a current graph. In at least some embodiments, users can be provided with an ability to control styles of generated road layouts mimicking existing cities, as well as to sketch a part of a road layout to be synthesized. Such a model can, in addition to synthesis, be used for tasks such as road parsing and path determination.

[0025] In at least one embodiment, a generative model for city road layouts can learn from available map data. One such model mimics turtle graphics methodology, in that road graphs can be grown progressively based on local statistics. For example, a city road layout can be modeled using a graph with nodes connected by edges as discussed above. In turtle graphics, a relative cursor is referred to as the turtle, which can move along a Cartesian plane. The turtle has a location and an orientation, and can move with commands relative to its position. As the turtle moves it creates a path, which can also have attributes such as width. An iterative, generative model can be thought of in at least one embodiment as moving one or more such turtles from each node, and making decisions for movement at each spatial control node.

[0026] A generative model in at least one embodiment can utilize an encoder-decoder architecture, where the encoder is a recurrent neural network (RNN) that encodes local incoming paths into a current node being analyzed, and the decoder is another RNN that generates outgoing nodes and edges connecting that current, existing node to the newly generated nodes. Generation is done iteratively, such as by pushing newly predicted nodes onto a queue, and finished once all nodes are visited. Such a model can be used to generate road layouts by additionally conditioning on a set of attributes, thus giving control to the user in generating the content. This model can, for example, take a user-specified partial sketch of the roads as input for generating a complete city (or region) road layout. Such a model can also be used as an effective prior for aerial map parsing, particularly in cases when the imagery varies in appearance from that used in training. Fine-tuning such a model jointly with a convolutional neural network (CNN)-based image feature extractor can further improves results, outperforming existing approaches. Various existing approaches only predict graph topology, and are unable to generate spatial graphs, as producing valid geometry and topology makes these tasks particularly challenging. In at least one embodiment, an encoder can learn node embeddings by encoding local connections using random walks. A model as discussed herein can utilize an encoder that operates directly on the graph, and a decoder that outputs several outgoing nodes using an RNN which may better capture more complex layout and path intersection topologies.

[0027] FIG. **1**A illustrates a pair of layouts **100** that can be generated in accordance with various embodiments. As illustrated, a first layout mimics a style of New York City streets, while a second layout mimics a style of London streets. It can be seen that these layouts have very distinct styles, which without such a process may require manual research and generation to model accurately and/or realistically. A model as discussed herein can extend or expand these layouts iteratively, making decisions on road directions, intersections, shapes, and other such aspects based on what has been learned for styles of the target location. In at least one embodiment a user may specify a city or location style, while in others a user can provide an initial sketch or graph portion from which such a style can be determined. FIG. **1**B illustrates a portion **150** of a third layout, which is generated with at least some amount of user (or other) interaction. In this example, features from different styles can be incorporated into a layout, such as here by specifying a New York style for one region of the layout, and a Cambridge style for another region of the layout. In some embodiments a user can specify to add a section of a specific style, or to change a portion of the layout to represent a certain style. In some embodiments the process will keep a queue of nodes for the different styles, and can add nodes and paths to the layout using the nodes from the various queues as appropriate. FIG. **2** illustrates a set **200** of aerial views of different locations, with different road layout styles. As illustrated, such a model can be used to iteratively generate a mapping of these regions, parsing the roads based upon captured image data. In at least one embodiment, the model can start at one or more locations of the aerial view data and iteratively grow the paths node-by-node as with other layout generation approaches discussed and suggested herein.

[0028] FIG. **3**A illustrates an example system **300** that can be used to generate layouts or graphs in accordance with various embodiments. In this example, a user can utilize a console **302** or other interface to provide input to a graph generation system **306**. The console can be any appropriate console, such as a computing device executing an interface application. The input provided can be any appropriate input as discussed and suggested herein, as may include a style to be used or a drawing of one or more portions of a layout to be generated. As mentioned, the input may also include image or map data, as may be pulled from a reference database **304** or other such location. The graph generation system may be an application running on the same device or infrastructure as the console, or may be executing on separate hardware that may be accessible over one or more networks. The graph generation system **306** can include a graph manager **308** that can take the input from the console and utilize a generative model to generate a graph or layout based at least in part upon that input. As mentioned, this may include an encoder RNN for encoding nodes of incoming paths to a current node, as well as a decoder RNN for decoding nodes for outgoing paths predicted for that current node. Once a final layout is generated in this example, that layout can be provided to the console **302** for presentation to a user, and can be stored to a graph database **314** locally or the reference database **304** for the client, among other such options.

[0029] In at least one embodiment, a vector graphics-based approach can be utilized for such a system, which may mimic functionality of turtle graphics as discussed above. In such an embodiment, a city road layout generation problem can be formulated as a planar graph generation problem. For example, a city road layout can be represented using an undirected graph $G=\{V,E\}$, with nodes V and edges E. A node $v_i \in V$ can encodes the location $[x_i, y_i]^T$ of the i^{th} control point, which could be an intersection, and an edge $E_{i,j} \in \{0, 1\}$ can denote whether a road segment connecting nodes v_i and v_j exists. An assumption can be made that the graph is a connected graph, such that a node can reach any other node by traversing a sequence of edges. The coordinates x_i's and y_i's can be defined relative to, for example, a world location of the city or other such location.

[0030] For a node v_i, an acyclic incoming path can be defined as an ordered sequence of unique, connected nodes which terminates at:

$$v_i: s^{in} = \{v_{i,1}, v_{i,2}, v_{i,3}, \ldots, v_{i,L}, v_i\}$$

where

$$e_{v_{i,t-1}, v_{i,t}} = 1 \text{ for each } 1 < t \leq L, \text{ and}$$

$$e_{v_{i,L}, v_i} = 1$$

with L representing the length of the path. It can be noted that multiple different acyclic paths can terminate at v_i, which can be denoted as $S_i^{in} := \{S_k^{in}\}$. Further, V_i^{out} can be defined as the set of all neighboring nodes of v_i, connected directly by an edge.

[0031] FIG. **3**B illustrates an example model **350** that can be utilized in accordance with various embodiments. This example depicts acyclic incoming paths $\{s^{in}\}$ to current (or "active") node **352** v_i. Each of these acyclic incoming paths, defined using neighboring nodes **354** within a determined proximity of the current node **352**, is encoded using an RNN

encoder **310**. A decoder RNN **312** of the model then uses this information to predict a set of predicted (or "outgoing") nodes **356** $\{v_{out}\}$.

[0032] In at least one embodiment, a generative model can generate a graph or layout in an iterative manner. In at least one embodiment, a queue of unvisited nodes can be maintained. At each iterative step, a node can be pulled from the queue and its local topology encoded. From there, a set of neighboring nodes can be decoded and these generated neighboring nodes can be pushed to the queue as additional nodes to be visited or analyzed for potential layout expansion. In at least one embodiment, this process can be repeated until the queue is empty, whereby all nodes have been visited, or until another end criterion is reached, such as where a target amount of expansion is reached or time has elapsed. In at least one embodiment, this graph can be initialized with a root node as well as its edges, providing a set of single connected, unvisited nodes with which the queue is initialized. Based on the way the graph is constructed, v_i has at least one neighbor node in the graph. The local topology can be represented using the set of acyclic incoming paths S_i^{in} to v_i (up to a maximum length L). These paths can be processed in an order-invariant manner and conditioned on the resulting latent representation to generate a set of outgoing nodes V_i^{out} (if any) and the edges connecting to them. FIG. **3**B provides a visualization of this process.

[0033] FIG. **4** illustrates a neural network architecture of an example generative model, including an encoder-decoder design. First, an encoder Gated Recurrent Unit (GRU) consumes the motion trajectory Δx^{in} of each incoming path of neighboring nodes **402**. An order-invariant representation can be produced by summing up the last-state hidden vectors across all paths. In this example, a decoder RNN then produces "commands" to advance the paths and produce new nodes, which can correspond to advancing the "turtle" in a turtle graphics-based approach. An optional attribute vector **404** can be further added to the decoder depending on factors such as the task to be performed.

[0034] In at least one embodiment, a single incoming path s^{in} into node v_i can be represented with a zero-initialized, bidirectional GRU (BGRU). Input to the BGRU at time step t can be the offset between two neighboring nodes:

$$v_{i,t}^{in} \in s^{in}$$

$$v_{i,t+1}^{in} \in s^{in}$$

in the path, such as maybe given by:

$$[\Delta x_{i,t}^{in}, \Delta y_{i,t}^{in}]^T = [x_{i,t+1}^{in}, y_{i,t+1}^{in}]^T - [x_{i,t}^{in}, y_{i,t}^{in}]^T$$

In at least one embodiment, each dimension can be encoded separately before feeding it into the GRU, which provides an ability to efficiently deal with high resolution map data.

[0035] Each node can have multiple incoming paths. In training, K random walks can be sampled, without repetition, where each random walk starts at v_i and can at most visit L different nodes. Such random sampling can lead to a more robust model as it covers more variants of a given graph. In at least one embodiment, each incoming path can be encoded individually, with a final latent representation h_{enc} being computed by summing the last hidden states of all paths. An attribute vector can additionally be appended to the latent representation. For example, the attribute could be a one-hot vector, encoding the city identity. This enables the generative model to learn an embedding of layout style, such as city road style, through utilizing additional linear layers to map the one-hot vector to the attribute vector h_{attr}. A final representation can then be given by $[h_{enc}, h_{attr}]$.

[0036] In at least one embodiment, a decoder of the model can decode the outgoing nodes V_i^{out} with a decoder GRU. The recurrent structure of the decoder can efficiently capture local dependencies between roads, such as orthogonality at road intersections. A process in accordance with at least one embodiment can independently predict $\Delta x_{t'}^{out}$ and $\Delta y_{t'}^{out}$ for an outgoing node $v_{t'}^{out}$, indicating a relative location of the node with respect to v_i. Such a process can additionally predict a binary variable which indicates whether to continue generating new nodes or stop. Optionally, a type can be predicted, such as the type of road between (v_i, v_t^{out}), such as a minor or major road, using a categorical variable. The hidden state $h_{t'}$ of the decoder can be updated as:

$$h_{t'+1} = GRU(h_{t'} | h_{enc}, h_{attr}, \Delta x_{t'}^{out})$$

[0037] As mentioned, such a process can be used for various applications, such as may include aerial road parsing. A dominant approach to parsing roads from aerial imagery has been to train a CNN to produce a probability map, followed by thresholding and thinning. However, this approach typically results in maps with holes or false positive road segments. Heuristics are then applied to post-process the topology. Approaches in accordance with various embodiments can train a generative model as described herein to generate graphs only on map data, in order to post-process the parsed topology. Starting from the most confident intersection node as the root node, all its neighbors can be pushed into the queue and the generative model used to expand the graph. At each decoding step, the road probability from CNN can be multiplied with the probability produced by NTG. An end of the decoder sequence is simply determined by checking whether the maximum probability falls below a threshold, such as 0.05.

[0038] For image-based layout generation, a generative model can be enabled to take image information into account by adding the CNN-predicted probabilities as input to the generative model NTG through the attribute vector, instead of applying the model only on the graph. In at least one embodiment, the encoder and decoder can remain the same. In practice, such a process can initialize the graph obtained by standard threshold and thinning, and an image-based generative model can be used to further refine the graph.

[0039] With respect to inferencing using such a generative model, a queue Q can be maintained. At each inference step, the first node can be pulled from the queue, its existing incoming paths encoded, and a set of new nodes generated. When a new node is produced, a check can be performed to determine whether the new node is proximal to an existing node in the graph. If the distance to an existing node is below a threshold ϵ (e.g., 5 m) then a new node does not need to be added to the queue. Instead, an edge is included to connect the current node to the existing node. This enables a graph to be created with cycles.

[0040] At training time, M incoming paths can be sampled for each v_i, with an aim to predict all of its neighboring nodes. An order can be enforced in decoding the nodes, where nodes can be sorted (for example and without limitation) a clockwise or counter-clockwise direction to form a sequence. Such an approach can help to avoid solving an assignment problem when computing the loss function. A

generative model can be trained using teacher-forcing with cross entropy loss for each of the output nodes. In at least one embodiment, 500 hidden dimensions can be used for both the encoder and decoder. The networks can be optimized using Adam with a learning rate of 1 e-3 and a weight decay rate of 1 e-4. In at least one embodiment, gradient clipping can also be applied with a threshold of, for example, 1.0.

[0041] In at least one embodiment, a single generative model architecture can be used for different tasks, such as city generation and road detection. Optimal parameterization strategy can vary depending at least in part upon the task to be performed. For example, discrete Δx, Δy with resolution of 1 m can be used for city generation for both encoder and decoder, where x points to east and y points to north. Here, Δx and Δy are limited to $[-100:100]$, indicating that the largest offset in either direction is 100 m. For road detection, continuous polar coordinates can be used in the encoder and discrete in the decoder, where the axis is rotated to align with the edge from the previous to the current node. This can help to form rotation-invariant motion trajectories that help detecting roads with arbitrary orientation. In at least one embodiment the coordinates x and y are encoded and predicted independently. Such an approach can yield similar results compared to predicting them jointly, while significantly saving training memory and model capacity.

[0042] In at least one embodiment, a generative model as discussed herein can be used for environment simulation. Such application can combine tasks discussed herein. For example, such a process can directly convert a satellite image into simulation-ready environments, which may be important for applications such as testing autonomous vehicle or robotic systems. In at least one embodiment, such a process can first detect roads in the satellite image with a generative model that provides an initial graph. This generative model can then be used to propose plausible variations. This can be performed in at least one embodiment by pushing all single-connection nodes in the initial parsing graph into a corresponding generative queue, and using the generative model to expand the graph until the queue is empty. Such an approach has various advantages. For example, it is fully automatic and only requires a low-cost satellite image as input. Second, it provides a set of plausible variations of the environment (e.g., city) instead of a static one, which could enable training more robust agents.

[0043] Such generative modeling can be performed in various other ways as well. For example, FIG. **5** illustrates a pair **400** of images that include a determined initial input region **406** to serve as a portion of the layout. The region **406** can be determined from a set of ground truth data **402** corresponding to an actual city layout. A generative model can use the nodes and edges of this region to generate an expanded layout **404** that uses the same style but produces a different overall layout, even with the same initial region. Such usage may find benefit in an application such as gaming, where there may be a core region that is intended to correspond to a recognizable portion of a city, but instead of having to accurately recreate the rest of the city a generative model can be used to generate a layout having the same or similar style, which will be fine for many users who may not be familiar with, or may not care about, the actual layout outside that core initial region. Such an application may also be useful for navigation and testing, where a specific layout portion may want to be retained for testing, such as to train on a difficult region or make sure a model does not forget what it has already learned, while also introducing new layouts of the target style. In some embodiments, a user may provide multiple such regions as a starting point, and a generative model can attempt to generate a layout that incorporates those regions as well as the styles of each. The local patterns in each of these provided regions can be remembered by the model and then intertwined to create novel structures and layouts.

[0044] FIG. **6** illustrates stages or steps **600** of layout generation in accordance with at least one embodiment. In this example, there are two different generations corresponding to A and B, where each includes initial input **602** provided by a user. In this example, the user can draw example intersections or road layout portions that can serve as a basis for the layout to be generated. This may have application in gaming or navigation, for example, where a user wants to have specific types of features in the layout to be generated. As illustrated, an iterative process can grow a layout from these inputs, using a style determined from the input (unless otherwise specified). In some embodiments, a user may be able to see the layout as it is generated and add additional features to help guide the layout.

[0045] FIG. **7** illustrates an example process **700** for generating a spatial layout that can be utilized in accordance with various embodiments. It should be understood for this and other processes discussed herein that there can be additional, alternative, or fewer steps performed in similar or alternative orders, or in parallel, within the scope of the various embodiments unless otherwise stated. In this example, an initial layout data is received **702** as input. As mentioned, this initial data can take many different forms, as may include an aerial image of a location, a map, a sketch, information indicating a particular style, or an initial layout region, among other such options. Based at least in part upon this input, an initial layout can be determined that includes one or more nodes that, for multiple nodes, are connected by one or more edges of the layout. As mentioned, such a process can grow this layout iteratively using a generative model, such that for each step a such a process can determine **704** nodes of the layout that have not yet been analyzed for potential path generation or layout expansion. As mentioned, this can include setting a variable indicating whether each node has been visited or storing those nodes to an analysis queue, among other such options. Once it is determined **706** that there are no more nodes to be visited, such as where there are no more nodes in a queue or all variables indicate the nodes have been visited, a final generated layout can be provided **708** as output of this process.

[0046] If there is at least one more node to be processed, a current node to be processed can be selected, such as by pulling that current node from a queue, and one or more neighboring nodes can be determined **710** that are within a determined distance of the current node. These nodes can be analyzed and encoded as representing incoming paths to the current node, such as by using an encoder RNN. As mentioned, such a process grows the layout locally, thus considering only neighboring nodes instead of an entire layout generated thus far. In this process, one or more styles to be used for generation from a current node can be determined **712**. In some embodiments this may be a single style determined at an earlier stage in the process, while in other embodiments this determination may be made for each current node, such as where there may be multiple styles

indicated for a layout and a determination is to be made as to which style, or styles, to utilize for the current node. Using the determined style(s) as well as the information from the neighboring nodes, one or more additional nodes can be predicted **714** that represent outgoing paths from that current node. This can be predicted using a decoder RNN in at least one embodiment. In at least one embodiment, an additional node is added only if the edge to that node would be of a minimum length. The new nodes can be flagged as nodes that have not yet been visited for expansion, such as by adding them to a queue or setting an appropriate visitation variable. The process can then continue for the next node, or set of nodes, until all nodes have been visited or an end criterion has been satisfied, among other such options.

[0047] Such a process can be thought of as working iteratively using local intersections. The layout can be expanded intersection by intersection, instead of trying to generate an entire graph or layout at once. These local patterns can be used to complete the entire graph iteratively, as encoding an entire graph in one pass may prove challenging or may provide sub-optimal results for at least some models or applications. Encoding locally, such as at a neighborhood level, is sufficient to capture the desired style, and portions of the layout can be decoded using local statistics. Such an approach can also be used to generate full high definition (HD) maps, including different types of roadways or paths instead of just segments indicating a general location of those roads or paths. Such an approach can also be used to fix or update HD maps in at least one embodiment.

[0048] FIG. **8** illustrates another example process **800** for generating a spatial layout that can be utilized in accordance with at least one embodiment. In this example, a first node of a spatial layout can be determined **802**. Neighboring nodes of the layout can be determined **804**, where those neighboring nodes are within a determined distance of the first node and represent incoming paths to the first node. These neighboring nodes can be decoded using a decoder RNN in at least one embodiment. A layout style to use for expanding the spatial layout can be determined **806**, and in some embodiments this may be determined before analyzing the first node. In some embodiments, different styles may be used for different nodes. In this example, the layout style and information about the neighboring nodes can be used to infer **808** at least one second node of the spatial layout. Each second node can represent a path segment from the first node in the spatial layout, and placement of each second node can be representative of the determined style. In at least one embodiment, these second nodes can be encoded using an encoder RNN. This process can be used iteratively to generate or expand a spatial layout, where each unvisited node can iteratively be treated as the first node until a final layout is generated.

Inference and Training Logic

[0049] FIG. **9**A illustrates inference and/or training logic **915** used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B.

[0050] In at least one embodiment, inference and/or training logic **915** may include, without limitation, code and/or data storage **901** 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 **915** may include, or be coupled to code and/or data storage **901** 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 **901** 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 **901** may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0051] In at least one embodiment, any portion of code and/or data storage **901** 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 **901** 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 **901** 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.

[0052] In at least one embodiment, inference and/or training logic **915** may include, without limitation, a code and/or data storage **905** 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 **905** 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 **915** may include, or be coupled to code and/or data storage **905** 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 **905** 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 **905** 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 **905** 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 **905** is internal or external to a processor, for example, or comprised of DRAM, SRAM, Flash or some other storage type may depend on available storage on-chip versus off-chip, latency requirements of training and/or inferencing functions being performed, batch size of data used in inferencing and/or training of a neural network, or some combination of these factors.

[0053] In at least one embodiment, code and/or data storage **901** and code and/or data storage **905** may be separate storage structures. In at least one embodiment, code and/or data storage **901** and code and/or data storage **905** may be same storage structure. In at least one embodiment, code and/or data storage **901** and code and/or data storage **905** may be partially same storage structure and partially separate storage structures. In at least one embodiment, any portion of code and/or data storage **901** and code and/or data storage **905** may be included with other on-chip or off-chip data storage, including a processor's L1, L2, or L3 cache or system memory.

[0054] In at least one embodiment, inference and/or training logic **915** may include, without limitation, one or more arithmetic logic unit(s) ("ALU(s)") **910**, 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 **920** that are functions of input/output and/or weight parameter data stored in code and/or data storage **901** and/or code and/or data storage **905**. In at least one embodiment, activations stored in activation storage **920** are generated according to linear algebraic and or matrix-based mathematics performed by ALU(s) **910** in response to performing instructions or other code, wherein weight values stored in code and/or data storage **905** and/or code and/or data storage **901** 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 **905** or code and/or data storage **901** or another storage on or off-chip.

[0055] In at least one embodiment, ALU(s) **910** are included within one or more processors or other hardware logic devices or circuits, whereas in another embodiment, ALU(s) **910** 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 **910** 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 **901**, code and/or data storage **905**, and activation storage **920** 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 **920** 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.

[0056] In at least one embodiment, activation storage **920** may be cache memory, DRAM, SRAM, non-volatile memory (e.g., Flash memory), or other storage. In at least one embodiment, activation storage **920** 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 **920** 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 **915** illustrated in FIG. **9**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 **915** illustrated in FIG. **9**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").

[0057] FIG. **9**B illustrates inference and/or training logic **915**, according to at least one or more embodiments. In at least one embodiment, inference and/or training logic **915** 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 **915** illustrated in FIG. **9**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 **915** illustrated in FIG. **9**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 **915** includes, without limitation, code and/or data storage **901** and code and/or data storage **905**, 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. **9**B, each of code and/or data storage **901** and code and/or data storage **905** is associated with a dedicated computational resource, such as computational hardware **902** and computational hardware **906**, respectively. In at least one embodiment, each of computational hardware **902** and computational hardware **906** comprises one or more ALUs that perform mathematical functions, such as linear algebraic functions, only on information stored in code

and/or data storage **901** and code and/or data storage **905**, respectively, result of which is stored in activation storage **920**.

[0058] In at least one embodiment, each of code and/or data storage **901** and **905** and corresponding computational hardware **902** and **906**, respectively, correspond to different layers of a neural network, such that resulting activation from one "storage/computational pair **901/902**" of code and/or data storage **901** and computational hardware **902** is provided as an input to "storage/computational pair **905/906**" of code and/or data storage **905** and computational hardware **906**, in order to mirror conceptual organization of a neural network. In at least one embodiment, each of storage/computational pairs **901/902** and **905/906** 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 **901/902** and **905/906** may be included in inference and/or training logic **915**.

Data Center

[0059] FIG. **10** illustrates an example data center **1000**, in which at least one embodiment may be used. In at least one embodiment, data center **1000** includes a data center infrastructure layer **1010**, a framework layer **1020**, a software layer **1030**, and an application layer **1040**.

[0060] In at least one embodiment, as shown in FIG. **10**, data center infrastructure layer **1010** may include a resource orchestrator **1012**, grouped computing resources **1014**, and node computing resources ("node C.R.s") **1016(1)-1016**(N), where "N" represents any whole, positive integer. In at least one embodiment, node C.R.s **1016(1)-1016**(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 **1016(1)-1016**(N) may be a server having one or more of above-mentioned computing resources.

[0061] In at least one embodiment, grouped computing resources **1014** 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 **1014** 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.

[0062] In at least one embodiment, resource orchestrator **1012** may configure or otherwise control one or more node C.R.s **1016(1)-1016**(N) and/or grouped computing resources **1014**. In at least one embodiment, resource orchestrator **1012** may include a software design infrastructure ("SDI") management entity for data center **1000**. In at least one embodiment, resource orchestrator may include hardware, software or some combination thereof.

[0063] In at least one embodiment, as shown in FIG. **10**, framework layer **1020** includes a job scheduler **1022**, a configuration manager **1024**, a resource manager **1026** and a distributed file system **1028**. In at least one embodiment, framework layer **1020** may include a framework to support software **1032** of software layer **1030** and/or one or more application(s) **1042** of application layer **1040**. In at least one embodiment, software **1032** or application(s) **1042** 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 **1020** 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 **1028** for large-scale data processing (e.g., "big data"). In at least one embodiment, job scheduler **1022** may include a Spark driver to facilitate scheduling of workloads supported by various layers of data center **1000**. In at least one embodiment, configuration manager **1024** may be capable of configuring different layers such as software layer **1030** and framework layer **1020** including Spark and distributed file system **1028** for supporting large-scale data processing. In at least one embodiment, resource manager **1026** may be capable of managing clustered or grouped computing resources mapped to or allocated for support of distributed file system **1028** and job scheduler **1022**. In at least one embodiment, clustered or grouped computing resources may include grouped computing resource **1014** at data center infrastructure layer **1010**. In at least one embodiment, resource manager **1026** may coordinate with resource orchestrator **1012** to manage these mapped or allocated computing resources.

[0064] In at least one embodiment, software **1032** included in software layer **1030** may include software used by at least portions of node C.R.s **1016(1)-1016**(N), grouped computing resources **1014**, and/or distributed file system **1028** of framework layer **1020**. 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.

[0065] In at least one embodiment, application(s) **1042** included in application layer **1040** may include one or more types of applications used by at least portions of node C.R.s **1016(1)-1016**(N), grouped computing resources **1014**, and/or distributed file system **1028** of framework layer **1020**. 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.

[0066] In at least one embodiment, any of configuration manager **1024**, resource manager **1026**, and resource orchestrator **1012** 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 **1000** from making possibly bad configuration decisions and possibly avoiding underutilized and/or poor performing portions of a data center.

[0067] In at least one embodiment, data center **1000** 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 **1000**. 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 **1000** by using weight parameters calculated through one or more training techniques described herein.

[0068] 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.

[0069] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** 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.

[0070] Components of these figures can be used to generate or expand spatial layouts. In particular, these components can be used with an iterative, generative neural network to produce layouts such as road layouts for cities and other such regions.

Computer Systems

[0071] FIG. **11** 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 **1100** 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 **1100** may include, without limitation, a component, such as a processor **1102** 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 **1100** 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 **1100** 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.

[0072] 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.

[0073] In at least one embodiment, computer system **1100** may include, without limitation, processor **1102** that may include, without limitation, one or more execution units **1108** to perform machine learning model training and/or inferencing according to techniques described herein. In at least one embodiment, computer system **1100** is a single processor desktop or server system, but in another embodiment computer system **1100** may be a multiprocessor system. In at least one embodiment, processor **1102** 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 **1102** may be coupled to a processor bus **1110** that may transmit data signals between processor **1102** and other components in computer system **1100**.

[0074] In at least one embodiment, processor **1102** may include, without limitation, a Level 1 ("L1") internal cache memory ("cache") **1104**. In at least one embodiment, processor **1102** may have a single internal cache or multiple levels of internal cache. In at least one embodiment, cache memory may reside external to processor **1102**. 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 **1106** may store different types of data in various registers including, without limitation, integer registers, floating point registers, status registers, and instruction pointer register.

[0075] In at least one embodiment, execution unit **1108**, including, without limitation, logic to perform integer and floating point operations, also resides in processor **1102**. In at least one embodiment, processor **1102** may also include a microcode ("ucode") read only memory ("ROM") that stores microcode for certain macro instructions. In at least one embodiment, execution unit **1108** may include logic to handle a packed instruction set **1109**. In at least one embodiment, by including packed instruction set **1109** in an instruction set of a general-purpose processor **1102**, along with associated circuitry to execute instructions, operations used by many multimedia applications may be performed using packed data in a general-purpose processor **1102**. 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.

[0076] In at least one embodiment, execution unit **1108** may also be used in microcontrollers, embedded processors, graphics devices, DSPs, and other types of logic circuits. In at least one embodiment, computer system **1100** may include, without limitation, a memory **1120**. In at least one embodiment, memory **1120** 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 **1120** may store instruction(s) **1119** and/or data **1121** represented by data signals that may be executed by processor **1102**.

[0077] In at least one embodiment, system logic chip may be coupled to processor bus **1110** and memory **1120**. In at least one embodiment, system logic chip may include, without limitation, a memory controller hub ("MCH") **1116**, and processor **1102** may communicate with MCH **1116** via processor bus **1110**. In at least one embodiment, MCH **1116** may provide a high bandwidth memory path **1118** to memory **1120** for instruction and data storage and for storage of graphics commands, data and textures. In at least one embodiment, MCH **1116** may direct data signals between processor **1102**, memory **1120**, and other components in computer system **1100** and to bridge data signals between processor bus **1110**, memory **1120**, and a system I/O **1122**. 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 **1116** may be coupled to memory **1120** through a high bandwidth memory path **1118** and graphics/video card **1112** may be coupled to MCH **1116** through an Accelerated Graphics Port ("AGP") interconnect **1114**.

[0078] In at least one embodiment, computer system **1100** may use system I/O **1122** that is a proprietary hub interface bus to couple MCH **1116** to I/O controller hub ("ICH") **1130**. In at least one embodiment, ICH **1130** 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 **1120**, chipset, and processor **1102**. Examples may include, without limitation, an audio controller **1129**, a firmware hub ("flash BIOS") **1128**, a wireless transceiver **1126**, a data storage **1124**, a legacy I/O controller **1123** containing user input and keyboard interfaces **1125**, a serial expansion port **1127**, such as Universal Serial Bus ("USB"), and a network controller **1134**. data storage **1124** may comprise a hard disk drive, a floppy disk drive, a CD-ROM device, a flash memory device, or other mass storage device.

[0079] In at least one embodiment, FIG. **11**A illustrates a system, which includes interconnected hardware devices or "chips", whereas in other embodiments, FIG. **11**A 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 **1100** are interconnected using compute express link (CXL) interconnects.

[0080] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** may be used in system FIG. **11**A 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.

[0081] Components of these figures can be used to generate or expand spatial layouts. In particular, these components can be used with an iterative, generative neural network to produce layouts such as road layouts for cities and other such regions.

[0082] FIG. **12** is a block diagram illustrating an electronic device **1200** for utilizing a processor **1210**, according to at least one embodiment. In at least one embodiment, electronic device **1200** 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.

[0083] In at least one embodiment, system **1200** may include, without limitation, processor **1210** communicatively coupled to any suitable number or kind of components, peripherals, modules, or devices. In at least one embodiment, processor **1210** 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. **12** illustrates a system, which includes interconnected hardware devices or "chips", whereas in other embodiments, FIG. **12** may illustrate an exemplary System on a Chip ("SoC"). In at least one embodiment, devices illustrated in FIG. **12** 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. **12** are interconnected using compute express link (CXL) interconnects.

[0084] In at least one embodiment, FIG. **12** may include a display **1224**, a touch screen **1225**, a touch pad **1230**, a Near Field Communications unit ("NFC") **1245**, a sensor hub **1240**, a thermal sensor **1246**, an Express Chipset ("EC") **1235**, a Trusted Platform Module ("TPM") **1238**, BIOS/firmware/flash memory ("BIOS, FW Flash") **1222**, a DSP **1260**, a drive **1220** such as a Solid State Disk ("SSD") or a Hard Disk Drive ("HDD"), a wireless local area network unit ("WLAN") **1250**, a Bluetooth unit **1252**, a Wireless Wide Area Network unit ("WWAN") **1256**, a Global Positioning System (GPS) **1255**, a camera ("USB 3.0 camera") **1254** such as a USB 3.0 camera, and/or a Low Power Double Data Rate ("LPDDR") memory unit ("LPDDR3") **1215** implemented in, for example, LPDDR3 standard. These components may each be implemented in any suitable manner.

[0085] In at least one embodiment, other components may be communicatively coupled to processor **1210** through components discussed above. In at least one embodiment, an accelerometer **1241**, Ambient Light Sensor ("ALS") **1242**, compass **1243**, and a gyroscope **1244** may be communicatively coupled to sensor hub **1240**. In at least one embodiment, thermal sensor **1239**, a fan **1237**, a keyboard **1246**, and a touch pad **1230** may be communicatively coupled to EC **1235**. In at least one embodiment, speaker **1263**, headphones

1264, and microphone ("mic") **1265** may be communicatively coupled to an audio unit ("audio codec and class d amp") **1262**, which may in turn be communicatively coupled to DSP **1260**. In at least one embodiment, audio unit **1264** 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") **1257** may be communicatively coupled to WWAN unit **1256**. In at least one embodiment, components such as WLAN unit **1250** and Bluetooth unit **1252**, as well as WWAN unit **1256** may be implemented in a Next Generation Form Factor ("NGFF").

[0086] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** may be used in system FIG. **12** 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.

[0087] Components of these figures can be used to generate or expand spatial layouts. In particular, these components can be used with an iterative, generative neural network to produce layouts such as road layouts for cities and other such regions.

[0088] FIG. **13** is a block diagram of a processing system, according to at least one embodiment. In at least one embodiment, system **1300** includes one or more processors **1302** and one or more graphics processors **1308**, and may be a single processor desktop system, a multiprocessor workstation system, or a server system having a large number of processors **1302** or processor cores **1307**. In at least one embodiment, system **1300** is a processing platform incorporated within a system-on-a-chip (SoC) integrated circuit for use in mobile, handheld, or embedded devices.

[0089] In at least one embodiment, system **1300** 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 **1300** is a mobile phone, smart phone, tablet computing device or mobile Internet device. In at least one embodiment, processing system **1300** 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 **1300** is a television or set top box device having one or more processors **1302** and a graphical interface generated by one or more graphics processors **1308**.

[0090] In at least one embodiment, one or more processors **1302** each include one or more processor cores **1307** 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 **1307** is configured to process a specific instruction set **1309**. In at least one embodiment, instruction set **1309** 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 **1307** may each process a different instruction set **1309**, which may include instructions to facilitate emulation of other instruction sets. In at least one embodiment, processor core **1307** may also include other processing devices, such a Digital Signal Processor (DSP).

[0091] In at least one embodiment, processor **1302** includes cache memory **1304**. In at least one embodiment, processor **1302** 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 **1302**. In at least one embodiment, processor **1302** 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 **1307** using known cache coherency techniques. In at least one embodiment, register file **1306** is additionally included in processor **1302** 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 **1306** may include general-purpose registers or other registers.

[0092] In at least one embodiment, one or more processor(s) **1302** are coupled with one or more interface bus(es) **1310** to transmit communication signals such as address, data, or control signals between processor **1302** and other components in system **1300**. In at least one embodiment, interface bus **1310**, 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 **1310** 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) **1302** include an integrated memory controller **1316** and a platform controller hub **1330**. In at least one embodiment, memory controller **1316** facilitates communication between a memory device and other components of system **1300**, while platform controller hub (PCH) **1330** provides connections to I/O devices via a local I/O bus.

[0093] In at least one embodiment, memory device **1320** 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 **1320** can operate as system memory for system **1300**, to store data **1322** and instructions **1321** for use when one or more processors **1302** executes an application or process. In at least one embodiment, memory controller **1316** also couples with an optional external graphics processor **1312**, which may communicate with one or more graphics processors **1308** in processors **1302** to perform graphics and media operations. In at least one embodiment, a display device **1311** can connect to processor(s) **1302**. In at least one embodiment display device **1311** 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 **1311** 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.

[0094] In at least one embodiment, platform controller hub **1330** enables peripherals to connect to memory device **1320** and processor **1302** via a high-speed I/O bus. In at least one embodiment, I/O peripherals include, but are not limited to, an audio controller **1346**, a network controller **1334**, a

firmware interface **1328**, a wireless transceiver **1326**, touch sensors **1325**, a data storage device **1324** (e.g., hard disk drive, flash memory, etc.). In at least one embodiment, data storage device **1324** 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 **1325** can include touch screen sensors, pressure sensors, or fingerprint sensors. In at least one embodiment, wireless transceiver **1326** 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 **1328** enables communication with system firmware, and can be, for example, a unified extensible firmware interface (UEFI). In at least one embodiment, network controller **1334** 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 **1310**. In at least one embodiment, audio controller **1346** is a multi-channel high definition audio controller. In at least one embodiment, system **1300** includes an optional legacy I/O controller **1340** for coupling legacy (e.g., Personal System 2 (PS/2)) devices to system. In at least one embodiment, platform controller hub **1330** can also connect to one or more Universal Serial Bus (USB) controllers **1342** connect input devices, such as keyboard and mouse **1343** combinations, a camera **1344**, or other USB input devices.

[0095] In at least one embodiment, an instance of memory controller **1316** and platform controller hub **1330** may be integrated into a discreet external graphics processor, such as external graphics processor **1312**. In at least one embodiment, platform controller hub **1330** and/or memory controller **1316** may be external to one or more processor(s) **1302**. For example, in at least one embodiment, system **1300** can include an external memory controller **1316** and platform controller hub **1330**, which may be configured as a memory controller hub and peripheral controller hub within a system chipset that is in communication with processor(s) **1302**.

[0096] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment portions or all of inference and/or training logic **915** may be incorporated into graphics processor **1300**. 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 **1312**. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIG. **9**A or **9**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 **1300** to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0097] Components of these figures can be used to generate or expand spatial layouts. In particular, these components can be used with an iterative, generative neural network to produce layouts such as road layouts for cities and other such regions.

[0098] FIG. **14** is a block diagram of a processor **1400** having one or more processor cores **1402**A-**1402**N, an integrated memory controller **1414**, and an integrated graphics processor **1408**, according to at least one embodiment. In at least one embodiment, processor **1400** can include additional cores up to and including additional core **1402**N represented by dashed lined boxes. In at least one embodiment, each of processor cores **1402**A-**1402**N includes one or more internal cache units **1404**A-**1404**N. In at least one embodiment, each processor core also has access to one or more shared cached units **1406**.

[0099] In at least one embodiment, internal cache units **1404**A-**1404**N and shared cache units **1406** represent a cache memory hierarchy within processor **1400**. In at least one embodiment, cache memory units **1404**A-**1404**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 **1406** and **1404**A-**1404**N.

[0100] In at least one embodiment, processor **1400** may also include a set of one or more bus controller units **1416** and a system agent core **1410**. In at least one embodiment, one or more bus controller units **1416** manage a set of peripheral buses, such as one or more PCI or PCI express busses. In at least one embodiment, system agent core **1410** provides management functionality for various processor components. In at least one embodiment, system agent core **1410** includes one or more integrated memory controllers **1414** to manage access to various external memory devices (not shown).

[0101] In at least one embodiment, one or more of processor cores **1402**A-**1402**N include support for simultaneous multi-threading. In at least one embodiment, system agent core **1410** includes components for coordinating and operating cores **1402**A-**1402**N during multi-threaded processing. In at least one embodiment, system agent core **1410** may additionally include a power control unit (PCU), which includes logic and components to regulate one or more power states of processor cores **1402**A-**1402**N and graphics processor **1408**.

[0102] In at least one embodiment, processor **1400** additionally includes graphics processor **1408** to execute graphics processing operations. In at least one embodiment, graphics processor **1408** couples with shared cache units **1406**, and system agent core **1410**, including one or more integrated memory controllers **1414**. In at least one embodiment, system agent core **1410** also includes a display controller **1411** to drive graphics processor output to one or more coupled displays. In at least one embodiment, display controller **1411** may also be a separate module coupled with graphics processor **1408** via at least one interconnect, or may be integrated within graphics processor **1408**.

[0103] In at least one embodiment, a ring based interconnect unit **1412** is used to couple internal components of processor **1400**. 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 **1408** couples with ring interconnect **1412** via an I/O link **1413**.

[0104] In at least one embodiment, I/O link **1413** 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 **1418**, such as an eDRAM module. In at least one embodiment, each of processor cores **1402**A-**1402**N and graphics processor **1408** use embedded memory modules **1418** as a shared Last Level Cache.

[0105] In at least one embodiment, processor cores **1402**A-**1402**N are homogenous cores executing a common instruction set architecture. In at least one embodiment, processor cores **1402**A-**1402**N are heterogeneous in terms of instruction set architecture (ISA), where one or more of processor cores **1402**A-**1402**N execute a common instruction set, while one or more other cores of processor cores **1402**A-**14**-**02**N executes a subset of a common instruction set or a different instruction set. In at least one embodiment, processor cores **1402**A-**1402**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 **1400** can be implemented on one or more chips or as an SoC integrated circuit.

[0106] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment portions or all of inference and/or training logic **915** may be incorporated into processor **1400**. 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 **1312**, graphics core(s) **1402**A-**1402**N, or other components in FIG. **14**. Moreover, in at least one embodiment, inferencing and/or training operations described herein may be done using logic other than logic illustrated in FIG. **9**A or **9**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 **1400** to perform one or more machine learning algorithms, neural network architectures, use cases, or training techniques described herein.

[0107] Components of these figures can be used to generate or expand spatial layouts. In particular, these components can be used with an iterative, generative neural network to produce layouts such as road layouts for cities and other such regions.

Autonomous Vehicle

[0108] FIG. **15**A illustrates an example of an autonomous vehicle **1500**, according to at least one embodiment. In at least one embodiment, autonomous vehicle **1500** (alternatively referred to herein as "vehicle **1500**") may be, without limitation, a passenger vehicle, such as a car, a truck, a bus, and/or another type of vehicle that accommodates one or more passengers. In at least one embodiment, vehicle **1*a*00** may be a semi-tractor-trailer truck used for hauling cargo. In at least one embodiment, vehicle **1*a*00** may be an airplane, robotic vehicle, or other kind of vehicle.

[0109] Autonomous vehicles may be described in terms of automation levels, defined by National Highway Traffic Safety Administration ("NHTSA"), a division of US Department of Transportation, and Society of Automotive Engineers ("SAE") "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles" (e.g., Standard No. J3016-201806, published on Jun. 15, 2018, Standard No. J3016-201609, published on Sep. 30, 2016, and previous and future versions of this standard). In one or more embodiments, vehicle **1500** may be capable of functionality in accordance with one or more of level 1-level 5 of autonomous driving levels. For example, in at least one embodiment, vehicle **1500** may be capable of conditional automation (Level 3), high automation (Level 4), and/or full automation (Level 5), depending on embodiment.

[0110] In at least one embodiment, vehicle **1500** may include, without limitation, components such as a chassis, a vehicle body, wheels (e.g., 2, 4, 6, 8, 18, etc.), tires, axles, and other components of a vehicle. In at least one embodiment, vehicle **1500** may include, without limitation, a propulsion system **1550**, such as an internal combustion engine, hybrid electric power plant, an all-electric engine, and/or another propulsion system type. In at least one embodiment, propulsion system **1550** may be connected to a drive train of vehicle **1500**, which may include, without limitation, a transmission, to enable propulsion of vehicle **1500**. In at least one embodiment, propulsion system **1550** may be controlled in response to receiving signals from a throttle/accelerator(s) **1552**.

[0111] In at least one embodiment, a steering system **1554**, which may include, without limitation, a steering wheel, is used to steer a vehicle **1500** (e.g., along a desired path or route) when a propulsion system **1550** is operating (e.g., when vehicle is in motion). In at least one embodiment, a steering system **1554** may receive signals from steering actuator(s) **1556**. A steering wheel may be optional for full automation (Level 5) functionality. In at least one embodiment, a brake sensor system **1546** may be used to operate vehicle brakes in response to receiving signals from brake actuator(s) **1548** and/or brake sensors.

[0112] In at least one embodiment, controller(s) **1536**, which may include, without limitation, one or more system on chips ("SoCs") (not shown in FIG. **15**A) and/or graphics processing unit(s) ("GPU(s)"), provide signals (e.g., representative of commands) to one or more components and/or systems of vehicle **1500**. For instance, in at least one embodiment, controller(s) **1536** may send signals to operate vehicle brakes via brake actuator(s) **1548**, to operate steering system **1554** via steering actuator(s) **1556**, and/or to operate propulsion system **1550** via throttle/accelerator(s) **1552**. Controller(s) **1536** may include one or more onboard (e.g., integrated) computing devices (e.g., supercomputers) that process sensor signals, and output operation commands (e.g., signals representing commands) to enable autonomous driving and/or to assist a human driver in driving vehicle **1500**. In at least one embodiment, controller(s) **1536** may include a first controller **1536** for autonomous driving functions, a second controller **1536** for functional safety functions, a third controller **1536** for artificial intelligence functionality (e.g., computer vision), a fourth controller **1536** for infotainment functionality, a fifth controller **1536** for redundancy in emergency conditions, and/or other controllers. In at least one embodiment, a single controller **1536** may handle two or more of above functionalities, two or more controllers **1536** may handle a single functionality, and/or any combination thereof.

[0113] In at least one embodiment, controller(s) **1536** provide signals for controlling one or more components and/or systems of vehicle **1500** in response to sensor data received from one or more sensors (e.g., sensor inputs). In

at least one embodiment, sensor data may be received from, for example and without limitation, global navigation satellite systems ("GNSS") sensor(s) **1558** (e.g., Global Positioning System sensor(s)), RADAR sensor(s) **1560**, ultrasonic sensor(s) **1562**, LIDAR sensor(s) **1564**, inertial measurement unit ("IMU") sensor(s) **1566** (e.g., accelerometer(s), gyroscope(s), magnetic compass(es), magnetometer (s), etc.), microphone(s) **1596**, stereo camera(s) **1568**, wide-view camera(s) **1570** (e.g., fisheye cameras), infrared camera(s) **1572**, surround camera(s) **1574** (e.g., 360 degree cameras), long-range cameras (not shown in FIG. **15**A), mid-range camera(s) (not shown in FIG. **15**A), speed sensor (s) **1544** (e.g., for measuring speed of vehicle **1500**), vibration sensor(s) **1542**, steering sensor(s) **1540**, brake sensor(s) (e.g., as part of brake sensor system **1546**), and/or other sensor types.

[0114] In at least one embodiment, one or more of controller(s) **1536** may receive inputs (e.g., represented by input data) from an instrument cluster **1532** of vehicle **1500** and provide outputs (e.g., represented by output data, display data, etc.) via a human-machine interface ("HMI") display **1534**, an audible annunciator, a loudspeaker, and/or via other components of vehicle **1500**. In at least one embodiment, outputs may include information such as vehicle velocity, speed, time, map data (e.g., a High Definition map (not shown in FIG. **15**A), location data (e.g., vehicle **1500**'s location, such as on a map), direction, location of other vehicles (e.g., an occupancy grid), information about objects and status of objects as perceived by controller(s) **1536**, etc. For example, in at least one embodiment, HMI display **1534** may display information about presence of one or more objects (e.g., a street sign, caution sign, traffic light changing, etc.), and/or information about driving maneuvers vehicle has made, is making, or will make (e.g., changing lanes now, taking exit **34**B in two miles, etc.).

[0115] In at least one embodiment, vehicle **1500** further includes a network interface **1524** which may use wireless antenna(s) **1526** and/or modem(s) to communicate over one or more networks. For example, in at least one embodiment, network interface **1524** may be capable of communication over Long-Term Evolution ("LTE"), Wideband Code Division Multiple Access ("WCDMA"), Universal Mobile Telecommunications System ("UMTS"), Global System for Mobile communication ("GSM"), IMT-CDMA Multi-Carrier ("CDMA2000"), etc. In at least one embodiment, wireless antenna(s) **1526** may also enable communication between objects in environment (e.g., vehicles, mobile devices, etc.), using local area network(s), such as Bluetooth, Bluetooth Low Energy ("LE"), Z-Wave, ZigBee, etc., and/or low power wide-area network(s) ("LPWANs"), such as LoRaWAN, SigFox, etc.

[0116] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** may be used in system FIG. **15**A 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.

[0117] Components of these figures can be used with layouts generated as discussed herein. In particular, these components can utilize road layouts for cities and other such regions for purposes such as navigation, testing, and training.

[0118] FIG. **15**B illustrates an example of camera locations and fields of view for autonomous vehicle **1500** of FIG. **15**A, according to at least one embodiment. In at least one embodiment, cameras and respective fields of view are one example embodiment and are not intended to be limiting. For instance, in at least one embodiment, additional and/or alternative cameras may be included and/or cameras may be located at different locations on vehicle **1500**.

[0119] In at least one embodiment, camera types for cameras may include, but are not limited to, digital cameras that may be adapted for use with components and/or systems of vehicle **1500**. In at least one embodiment, one or more of camera(s) may operate at automotive safety integrity level ("ASIL") B and/or at another ASIL. In at least one embodiment, camera types may be capable of any image capture rate, such as 60 frames per second (fps), 120 fps, 240 fps, etc., depending on embodiment. In at least one embodiment, cameras may be capable of using rolling shutters, global shutters, another type of shutter, or a combination thereof. In at least one embodiment, color filter array may include a red clear clear clear ("RCCC") color filter array, a red clear clear blue ("RCCB") color filter array, a red blue green clear ("RBGC") color filter array, a Foveon X3 color filter array, a Bayer sensors ("RGGB") color filter array, a monochrome sensor color filter array, and/or another type of color filter array. In at least one embodiment, clear pixel cameras, such as cameras with an RCCC, an RCCB, and/or an RBGC color filter array, may be used in an effort to increase light sensitivity.

[0120] In at least one embodiment, one or more of camera (s) may be used to perform advanced driver assistance systems ("ADAS") functions (e.g., as part of a redundant or fail-safe design). For example, in at least one embodiment, a Multi-Function Mono Camera may be installed to provide functions including lane departure warning, traffic sign assist and intelligent headlamp control. In at least one embodiment, one or more of camera(s) (e.g., all of cameras) may record and provide image data (e.g., video) simultaneously.

[0121] In at least one embodiment, one or more of cameras may be mounted in a mounting assembly, such as a custom designed (three-dimensional ("3D") printed) assembly, in order to cut out stray light and reflections from within car (e.g., reflections from dashboard reflected in windshield mirrors) which may interfere with camera's image data capture abilities. With reference to wing-mirror mounting assemblies, in at least one embodiment, wing-mirror assemblies may be custom 3D printed so that camera mounting plate matches shape of wing-mirror. In at least one embodiment, camera(s) may be integrated into wing-mirror. For side-view cameras, camera(s) may also be integrated within four pillars at each corner of cabIn at least one embodiment.

[0122] In at least one embodiment, cameras with a field of view that include portions of environment in front of vehicle **1500** (e.g., front-facing cameras) may be used for surround view, to help identify forward facing paths and obstacles, as well as aid in, with help of one or more of controllers **1536** and/or control SoCs, providing information critical to generating an occupancy grid and/or determining preferred vehicle paths. In at least one embodiment, front-facing cameras may be used to perform many of same ADAS

functions as LIDAR, including, without limitation, emergency braking, pedestrian detection, and collision avoidance. In at least one embodiment, front-facing cameras may also be used for ADAS functions and systems including, without limitation, Lane Departure Warnings ("LDW"), Autonomous Cruise Control ("ACC"), and/or other functions such as traffic sign recognition.

[0123] In at least one embodiment, a variety of cameras may be used in a front-facing configuration, including, for example, a monocular camera platform that includes a CMOS ("complementary metal oxide semiconductor") color imager. In at least one embodiment, wide-view camera **1570** may be used to perceive objects coming into view from periphery (e.g., pedestrians, crossing traffic or bicycles). Although only one wide-view camera **1570** is illustrated in FIG. **15**B, in other embodiments, there may be any number (including zero) of wide-view camera(s) **1570** on vehicle **1500**. In at least one embodiment, any number of long-range camera(s) **1598** (e.g., a long-view stereo camera pair) may be used for depth-based object detection, especially for objects for which a neural network has not yet been trained. In at least one embodiment, long-range camera(s) **1598** may also be used for object detection and classification, as well as basic object tracking.

[0124] In at least one embodiment, any number of stereo camera(s) **1568** may also be included in a front-facing configuration. In at least one embodiment, one or more of stereo camera(s) **1568** may include an integrated control unit comprising a scalable processing unit, which may provide a programmable logic ("FPGA") and a multi-core microprocessor with an integrated Controller Area Network ("CAN") or Ethernet interface on a single chip. In at least one embodiment, such a unit may be used to generate a 3D map of environment of vehicle **1500**, including a distance estimate for all points in image. In at least one embodiment, one or more of stereo camera(s) **1568** may include, without limitation, compact stereo vision sensor(s) that may include, without limitation, two camera lenses (one each on left and right) and an image processing chip that may measure distance from vehicle **1500** to target object and use generated information (e.g., metadata) to activate autonomous emergency braking and lane departure warning functions. In at least one embodiment, other types of stereo camera(s) **1568** may be used in addition to, or alternatively from, those described herein.

[0125] In at least one embodiment, cameras with a field of view that include portions of environment to side of vehicle **1500** (e.g., side-view cameras) may be used for surround view, providing information used to create and update occupancy grid, as well as to generate side impact collision warnings. For example, in at least one embodiment, surround camera(s) **1574** (e.g., four surround cameras **1574** as illustrated in FIG. **15**B) could be positioned on vehicle **1500**. In at least one embodiment, surround camera(s) **1574** may include, without limitation, any number and combination of wide-view camera(s) **1570**, fisheye camera(s), 360 degree camera(s), and/or like. For instance, in at least one embodiment, four fisheye cameras may be positioned on front, rear, and sides of vehicle **1500**. In at least one embodiment, vehicle **1500** may use three surround camera(s) **1574** (e.g., left, right, and rear), and may leverage one or more other camera(s) (e.g., a forward-facing camera) as a fourth surround-view camera.

[0126] In at least one embodiment, cameras with a field of view that include portions of environment to rear of vehicle **1500** (e.g., rear-view cameras) may be used for park assistance, surround view, rear collision warnings, and creating and updating occupancy grid. In at least one embodiment, a wide variety of cameras may be used including, but not limited to, cameras that are also suitable as a front-facing camera(s) (e.g., long-range cameras **1598** and/or mid-range camera(s) **1576**, stereo camera(s) **1568**), infrared camera(s) **1572**, etc.), as described herein.

[0127] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** may be used in system FIG. **15**B 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.

[0128] Components of these figures can be used with layouts generated as discussed herein. In particular, these components can utilize road layouts for cities and other such regions for purposes such as navigation, testing, and training.

[0129] FIG. **15**C is a block diagram illustrating an example system architecture for autonomous vehicle **1500** of FIG. **15**A, according to at least one embodiment. In at least one embodiment, each of components, features, and systems of vehicle **1500** in FIG. **15**C are illustrated as being connected via a bus **1502**. In at least one embodiment, bus **1502** may include, without limitation, a CAN data interface (alternatively referred to herein as a "CAN bus"). In at least one embodiment, a CAN bus may be a network inside vehicle **1500** used to aid in control of various features and functionality of vehicle **1500**, such as actuation of brakes, acceleration, braking, steering, windshield wipers, etc. In at least one embodiment, bus **1502** may be configured to have dozens or even hundreds of nodes, each with its own unique identifier (e.g., a CAN ID). In at least one embodiment, bus **1502** may be read to find steering wheel angle, ground speed, engine revolutions per minute ("RPMs"), button positions, and/or other vehicle status indicators. In at least one embodiment, bus **1502** may be a CAN bus that is ASIL B compliant.

[0130] In at least one embodiment, in addition to, or alternatively from CAN, FlexRay and/or Ethernet may be used. In at least one embodiment, there may be any number of busses **1502**, which may include, without limitation, zero or more CAN busses, zero or more FlexRay busses, zero or more Ethernet busses, and/or zero or more other types of busses using a different protocol. In at least one embodiment, two or more busses **1502** may be used to perform different functions, and/or may be used for redundancy. For example, a first bus **1502** may be used for collision avoidance functionality and a second bus **1502** may be used for actuation control. In at least one embodiment, each bus **1502** may communicate with any of components of vehicle **1500**, and two or more busses **1502** may communicate with same components. In at least one embodiment, each of any number of system(s) on chip(s) ("SoC(s)") **1504**, each of controller(s) **1536**, and/or each computer within vehicle may

have access to same input data (e.g., inputs from sensors of vehicle **1500**), and may be connected to a common bus, such CAN bus.

[0131] In at least one embodiment, vehicle **1500** may include one or more controller(s) **1536**, such as those described herein with respect to FIG. **15**A. Controller(s) **1536** may be used for a variety of functions. In at least one embodiment, controller(s) **1536** may be coupled to any of various other components and systems of vehicle **1500**, and may be used for control of vehicle **1500**, artificial intelligence of vehicle **1500**, infotainment for vehicle **1500**, and/or like.

[0132] In at least one embodiment, vehicle **1500** may include any number of SoCs **1504**. Each of SoCs **1504** may include, without limitation, central processing units ("CPU(s)") **1506**, graphics processing units ("GPU(s)") **1508**, processor(s) **1510**, cache(s) **1512**, accelerator(s) **1514**, data store(s) **1516**, and/or other components and features not illustrated. In at least one embodiment, SoC(s) **1504** may be used to control vehicle **1500** in a variety of platforms and systems. For example, in at least one embodiment, SoC(s) **1504** may be combined in a system (e.g., system of vehicle **1500**) with a High Definition ("HD") map **1522** which may obtain map refreshes and/or updates via network interface **1524** from one or more servers (not shown in FIG. **15**C).

[0133] In at least one embodiment, CPU(s) **1506** may include a CPU cluster or CPU complex (alternatively referred to herein as a "CCPLEX"). In at least one embodiment, CPU(s) **1506** may include multiple cores and/or level two ("L2") caches. For instance, in at least one embodiment, CPU(s) **1506** may include eight cores in a coherent multi-processor configuration. In at least one embodiment, CPU(s) **1506** may include four dual-core clusters where each cluster has a dedicated L2 cache (e.g., a 2 MB L2 cache). In at least one embodiment, CPU(s) **1506** (e.g., CCPLEX) may be configured to support simultaneous cluster operation enabling any combination of clusters of CPU(s) **1506** to be active at any given time.

[0134] In at least one embodiment, one or more of CPU(s) **1506** may implement power management capabilities that include, without limitation, one or more of following features: individual hardware blocks may be clock-gated automatically when idle to save dynamic power; each core clock may be gated when core is not actively executing instructions due to execution of Wait for Interrupt ("WFI")/Wait for Event ("WFE") instructions; each core may be independently power-gated; each core cluster may be independently clock-gated when all cores are clock-gated or power-gated; and/or each core cluster may be independently power-gated when all cores are power-gated. In at least one embodiment, CPU(s) **1506** may further implement an enhanced algorithm for managing power states, where allowed power states and expected wakeup times are specified, and hardware/microcode determines best power state to enter for core, cluster, and CCPLEX. In at least one embodiment, processing cores may support simplified power state entry sequences in software with work offloaded to microcode.

[0135] In at least one embodiment, GPU(s) **1508** may include an integrated GPU (alternatively referred to herein as an "iGPU"). In at least one embodiment, GPU(s) **1508** may be programmable and may be efficient for parallel workloads. In at least one embodiment, GPU(s) **1508**, in at least one embodiment, may use an enhanced tensor instruction set. In at least one embodiment, GPU(s) **1508** may include one or more streaming microprocessors, where each streaming microprocessor may include a level one ("L1") cache (e.g., an L1 cache with at least 96 KB storage capacity), and two or more of streaming microprocessors may share an L2 cache (e.g., an L2 cache with a 512 KB storage capacity). In at least one embodiment, GPU(s) **1508** may include at least eight streaming microprocessors. In at least one embodiment, GPU(s) **1508** may use compute application programming interface(s) (API(s)). In at least one embodiment, GPU(s) **1508** may use one or more parallel computing platforms and/or programming models (e.g., NVIDIA's CUDA).

[0136] In at least one embodiment, one or more of GPU(s) **1508** may be power-optimized for best performance in automotive and embedded use cases. For example, in on embodiment, GPU(s) **1508** could be fabricated on a Fin field-effect transistor ("FinFET"). In at least one embodiment, each streaming microprocessor may incorporate a number of mixed-precision processing cores partitioned into multiple blocks. For example, and without limitation, 64 PF32 cores and 32 PF64 cores could be partitioned into four processing blocks. In at least one embodiment, each processing block could be allocated 16 FP32 cores, 8 FP64 cores, 16 INT32 cores, two mixed-precision NVIDIA TENSOR CORES for deep learning matrix arithmetic, a level zero ("L0") instruction cache, a warp scheduler, a dispatch unit, and/or a 64 KB register file. In at least one embodiment, streaming microprocessors may include independent parallel integer and floating-point data paths to provide for efficient execution of workloads with a mix of computation and addressing calculations. In at least one embodiment, streaming microprocessors may include independent thread scheduling capability to enable finer-grain synchronization and cooperation between parallel threads. In at least one embodiment, streaming microprocessors may include a combined L1 data cache and shared memory unit in order to improve performance while simplifying programming.

[0137] In at least one embodiment, one or more of GPU(s) **1508** may include a high bandwidth memory ("HBM) and/or a 16 GB HBM2 memory subsystem to provide, in some examples, about 900 GB/second peak memory bandwidth. In at least one embodiment, in addition to, or alternatively from, HBM memory, a synchronous graphics random-access memory ("SGRAM") may be used, such as a graphics double data rate type five synchronous random-access memory ("GDDR5").

[0138] In at least one embodiment, GPU(s) **1508** may include unified memory technology. In at least one embodiment, address translation services ("ATS") support may be used to allow GPU(s) **1508** to access CPU(s) **1506** page tables directly. In at least one embodiment, embodiment, when GPU(s) **1508** memory management unit ("MMU") experiences a miss, an address translation request may be transmitted to CPU(s) **1506**. In response, CPU(s) **1506** may look in its page tables for virtual-to-physical mapping for address and transmits translation back to GPU(s) **1508**, in at least one embodiment. In at least one embodiment, unified memory technology may allow a single unified virtual address space for memory of both CPU(s) **1506** and GPU(s) **1508**, thereby simplifying GPU(s) **1508** programming and porting of applications to GPU(s) **1508**.

[0139] In at least one embodiment, GPU(s) **1508** may include any number of access counters that may keep track of frequency of access of GPU(s) **1508** to memory of other

processors. In at least one embodiment, access counter(s) may help ensure that memory pages are moved to physical memory of processor that is accessing pages most frequently, thereby improving efficiency for memory ranges shared between processors.

[0140] In at least one embodiment, one or more of SoC(s) **1504** may include any number of cache(s) **1512**, including those described herein. For example, in at least one embodiment, cache(s) **1512** could include a level three ("L3") cache that is available to both CPU(s) **1506** and GPU(s) **1508** (e.g., that is connected both CPU(s) **1506** and GPU(s) **1508**). In at least one embodiment, cache(s) **1512** may include a write-back cache that may keep track of states of lines, such as by using a cache coherence protocol (e.g., MEI, MESI, MSI, etc.). In at least one embodiment, L3 cache may include 4 MB or more, depending on embodiment, although smaller cache sizes may be used.

[0141] In at least one embodiment, one or more of SoC(s) **1504** may include one or more accelerator(s) **1514** (e.g., hardware accelerators, software accelerators, or a combination thereof). In at least one embodiment, SoC(s) **1504** may include a hardware acceleration cluster that may include optimized hardware accelerators and/or large on-chip memory. In at least one embodiment, large on-chip memory (e.g., 4 MB of SRAM), may enable hardware acceleration cluster to accelerate neural networks and other calculations. In at least one embodiment, hardware acceleration cluster may be used to complement GPU(s) **1508** and to off-load some of tasks of GPU(s) **1508** (e.g., to free up more cycles of GPU(s) **1508** for performing other tasks). In at least one embodiment, accelerator(s) **1514** could be used for targeted workloads (e.g., perception, convolutional neural networks ("CNNs"), recurrent neural networks ("RNNs"), etc.) that are stable enough to be amenable to acceleration. In at least one embodiment, a CNN may include a region-based or regional convolutional neural networks ("RCNNs") and Fast RCNNs (e.g., as used for object detection) or other type of CNN.

[0142] In at least one embodiment, accelerator(s) **1514** (e.g., hardware acceleration cluster) may include a deep learning accelerator(s) ("DLA(s)"). DLA(s) may include, without limitation, one or more Tensor processing units ("TPU(s)") that may be configured to provide an additional ten trillion operations per second for deep learning applications and inferencing. In at least one embodiment, TPU(s) may be accelerators configured to, and optimized for, performing image processing functions (e.g., for CNNs, RCNNs, etc.). DLA(s) may further be optimized for a specific set of neural network types and floating point operations, as well as inferencing. In at least one embodiment, design of DLA(s) may provide more performance per millimeter than a typical general-purpose GPU, and typically vastly exceeds performance of a CPU. In at least one embodiment, TPU(s) may perform several functions, including a single-instance convolution function, supporting, for example, INT8, INT16, and FP16 data types for both features and weights, as well as post-processor functions. In at least one embodiment, DLA(s) may quickly and efficiently execute neural networks, especially CNNs, on processed or unprocessed data for any of a variety of functions, including, for example and without limitation: a CNN for object identification and detection using data from camera sensors; a CNN for distance estimation using data from camera sensors; a CNN for emergency vehicle detection and identification and detection using data from microphones **1596**; a CNN for facial recognition and vehicle owner identification using data from camera sensors; and/or a CNN for security and/or safety related events.

[0143] In at least one embodiment, DLA(s) may perform any function of GPU(s) **1508**, and by using an inference accelerator, for example, a designer may target either DLA(s) or GPU(s) **1508** for any function. For example, in at least one embodiment, designer may focus processing of CNNs and floating point operations on DLA(s) and leave other functions to GPU(s) **1508** and/or other accelerator(s) **1514**.

[0144] In at least one embodiment, accelerator(s) **1514** (e.g., hardware acceleration cluster) may include a programmable vision accelerator(s) ("PVA"), which may alternatively be referred to herein as a computer vision accelerator. In at least one embodiment, PVA(s) may be designed and configured to accelerate computer vision algorithms for advanced driver assistance system ("ADAS") **1538**, autonomous driving, augmented reality ("AR") applications, and/or virtual reality ("VR") applications. PVA(s) may provide a balance between performance and flexibility. For example, in at least one embodiment, each PVA(s) may include, for example and without limitation, any number of reduced instruction set computer ("RISC") cores, direct memory access ("DMA"), and/or any number of vector processors.

[0145] In at least one embodiment, RISC cores may interact with image sensors (e.g., image sensors of any of cameras described herein), image signal processor(s), and/or like. In at least one embodiment, each of RISC cores may include any amount of memory. In at least one embodiment, RISC cores may use any of a number of protocols, depending on embodiment. In at least one embodiment, RISC cores may execute a real-time operating system ("RTOS"). In at least one embodiment, RISC cores may be implemented using one or more integrated circuit devices, application specific integrated circuits ("ASICs"), and/or memory devices. For example, in at least one embodiment, RISC cores could include an instruction cache and/or a tightly coupled RAM.

[0146] In at least one embodiment, DMA may enable components of PVA(s) to access system memory independently of CPU(s) **1506**. In at least one embodiment, DMA may support any number of features used to provide optimization to PVA including, but not limited to, supporting multi-dimensional addressing and/or circular addressing. In at least one embodiment, DMA may support up to six or more dimensions of addressing, which may include, without limitation, block width, block height, block depth, horizontal block stepping, vertical block stepping, and/or depth stepping.

[0147] In at least one embodiment, vector processors may be programmable processors that may be designed to efficiently and flexibly execute programming for computer vision algorithms and provide signal processing capabilities. In at least one embodiment, PVA may include a PVA core and two vector processing subsystem partitions. In at least one embodiment, PVA core may include a processor subsystem, DMA engine(s) (e.g., two DMA engines), and/or other peripherals. In at least one embodiment, vector processing subsystem may operate as primary processing engine of PVA, and may include a vector processing unit ("VPU"), an instruction cache, and/or vector memory (e.g., "VMEM"). In at least one embodiment, VPU may include a digital signal processor such as, for example, a single

instruction, multiple data ("SIMD"), very long instruction word ("VLIW") digital signal processor. In at least one embodiment, a combination of SIMD and VLIW may enhance throughput and speed.

[0148] In at least one embodiment, each of vector processors may include an instruction cache and may be coupled to dedicated memory. As a result, in at least one embodiment, each of vector processors may be configured to execute independently of other vector processors. In at least one embodiment, vector processors that are included in a particular PVA may be configured to employ data parallelism. For instance, in at least one embodiment, plurality of vector processors included in a single PVA may execute same computer vision algorithm, but on different regions of an image. In at least one embodiment, vector processors included in a particular PVA may simultaneously execute different computer vision algorithms, on same image, or even execute different algorithms on sequential images or portions of an image. In at least one embodiment, among other things, any number of PVAs may be included in hardware acceleration cluster and any number of vector processors may be included in each of PVAs. In at least one embodiment, PVA(s) may include additional error correcting code ("ECC") memory, to enhance overall system safety.

[0149] In at least one embodiment, accelerator(s) **1514** (e.g., hardware acceleration cluster) may include a computer vision network on-chip and static random-access memory ("SRAM"), for providing a high-bandwidth, low latency SRAM for accelerator(s) **1514**. In at least one embodiment, on-chip memory may include at least 4 MB SRAM, consisting of, for example and without limitation, eight field-configurable memory blocks, that may be accessible by both PVA and DLA. In at least one embodiment, each pair of memory blocks may include an advanced peripheral bus ("APB") interface, configuration circuitry, a controller, and a multiplexer. In at least one embodiment, any type of memory may be used. In at least one embodiment, PVA and DLA may access memory via a backbone that provides PVA and DLA with high-speed access to memory. In at least one embodiment, backbone may include a computer vision network on-chip that interconnects PVA and DLA to memory (e.g., using APB).

[0150] In at least one embodiment, computer vision network on-chip may include an interface that determines, before transmission of any control signal/address/data, that both PVA and DLA provide ready and valid signals. In at least one embodiment, an interface may provide for separate phases and separate channels for transmitting control signals/addresses/data, as well as burst-type communications for continuous data transfer. In at least one embodiment, an interface may comply with International Organization for Standardization ("ISO") **26262** or International Electrotechnical Commission ("IEC") 61508 standards, although other standards and protocols may be used.

[0151] In at least one embodiment, one or more of SoC(s) **1504** may include a real-time ray-tracing hardware accelerator. In at least one embodiment, real-time ray-tracing hardware accelerator may be used to quickly and efficiently determine positions and extents of objects (e.g., within a world model), to generate real-time visualization simulations, for RADAR signal interpretation, for sound propagation synthesis and/or analysis, for simulation of SONAR systems, for general wave propagation simulation, for comparison to LIDAR data for purposes of localization and/or other functions, and/or for other uses.

[0152] In at least one embodiment, accelerator(s) **1514** (e.g., hardware accelerator cluster) have a wide array of uses for autonomous driving. In at least one embodiment, PVA may be a programmable vision accelerator that may be used for key processing stages in ADAS and autonomous vehicles. In at least one embodiment, PVA's capabilities are a good match for algorithmic domains needing predictable processing, at low power and low latency. In other words, PVA performs well on semi-dense or dense regular computation, even on small data sets, which need predictable run-times with low latency and low power. In at least one embodiment, autonomous vehicles, such as vehicle **1500**, PVAs are designed to run classic computer vision algorithms, as they are efficient at object detection and operating on integer math.

[0153] For example, according to at least one embodiment of technology, PVA is used to perform computer stereo vision. In at least one embodiment, semi-global matching-based algorithm may be used in some examples, although this is not intended to be limiting. In at least one embodiment, applications for Level 3-5 autonomous driving use motion estimation/stereo matching on-the-fly (e.g., structure from motion, pedestrian recognition, lane detection, etc.). In at least one embodiment, PVA may perform computer stereo vision function on inputs from two monocular cameras.

[0154] In at least one embodiment, PVA may be used to perform dense optical flow. For example, in at least one embodiment, PVA could process raw RADAR data (e.g., using a 4D Fast Fourier Transform) to provide processed RADAR data. In at least one embodiment, PVA is used for time of flight depth processing, by processing raw time of flight data to provide processed time of flight data, for example.

[0155] In at least one embodiment, DLA may be used to run any type of network to enhance control and driving safety, including for example and without limitation, a neural network that outputs a measure of confidence for each object detection. In at least one embodiment, confidence may be represented or interpreted as a probability, or as providing a relative "weight" of each detection compared to other detections. In at least one embodiment, confidence enables a system to make further decisions regarding which detections should be considered as true positive detections rather than false positive detections. For example, In at least one embodiment, a system may set a threshold value for confidence and consider only detections exceeding threshold value as true positive detections. In an embodiment in which an automatic emergency braking ("AEB") system is used, false positive detections would cause vehicle to automatically perform emergency braking, which is obviously undesirable. In at least one embodiment, highly confident detections may be considered as triggers for AEB. In at least one embodiment, DLA may run a neural network for regressing confidence value. In at least one embodiment, neural network may take as its input at least some subset of parameters, such as bounding box dimensions, ground plane estimate obtained (e.g. from another subsystem), output from IMU sensor(s) **1566** that correlates with vehicle **1500** orientation, distance, 3D location estimates of object obtained from neural network and/or other sensors (e.g., LIDAR sensor(s) **1564** or RADAR sensor(s) **1560**), among others.

[0156] In at least one embodiment, one or more of SoC(s) **1504** may include data store(s) **1516** (e.g., memory). In at least one embodiment, data store(s) **1516** may be on-chip memory of SoC(s) **1504**, which may store neural networks to be executed on GPU(s) **1508** and/or DLA. In at least one embodiment, data store(s) **1516** may be large enough in capacity to store multiple instances of neural networks for redundancy and safety. In at least one embodiment, data store(s) **1516** may comprise L2 or L3 cache(s).

[0157] In at least one embodiment, one or more of SoC(s) **1504** may include any number of processor(s) **1510** (e.g., embedded processors). In at least one embodiment, processor(s) **1510** may include a boot and power management processor that may be a dedicated processor and subsystem to handle boot power and management functions and related security enforcement. In at least one embodiment, boot and power management processor may be a part of SoC(s) **1504** boot sequence and may provide runtime power management services. In at least one embodiment, boot power and management processor may provide clock and voltage programming, assistance in system low power state transitions, management of SoC(s) **1504** thermals and temperature sensors, and/or management of SoC(s) **1504** power states. In at least one embodiment, each temperature sensor may be implemented as a ring-oscillator whose output frequency is proportional to temperature, and SoC(s) **1504** may use ring-oscillators to detect temperatures of CPU(s) **1506**, GPU(s) **1508**, and/or accelerator(s) **1514**. In at least one embodiment, if temperatures are determined to exceed a threshold, then boot and power management processor may enter a temperature fault routine and put SoC(s) **1504** into a lower power state and/or put vehicle **1500** into a chauffeur to safe stop mode (e.g., bring vehicle **1500** to a safe stop).

[0158] In at least one embodiment, processor(s) **1510** may further include a set of embedded processors that may serve as an audio processing engine. In at least one embodiment, audio processing engine may be an audio subsystem that enables full hardware support for multi-channel audio over multiple interfaces, and a broad and flexible range of audio I/O interfaces. In at least one embodiment, audio processing engine is a dedicated processor core with a digital signal processor with dedicated RAM.

[0159] In at least one embodiment, processor(s) **1510** may further include an always on processor engine that may provide necessary hardware features to support low power sensor management and wake use cases. In at least one embodiment, always on processor engine may include, without limitation, a processor core, a tightly coupled RAM, supporting peripherals (e.g., timers and interrupt controllers), various I/O controller peripherals, and routing logic.

[0160] In at least one embodiment, processor(s) **1510** may further include a safety cluster engine that includes, without limitation, a dedicated processor subsystem to handle safety management for automotive applications. In at least one embodiment, safety cluster engine may include, without limitation, two or more processor cores, a tightly coupled RAM, support peripherals (e.g., timers, an interrupt controller, etc.), and/or routing logic. In a safety mode, two or more cores may operate, in at least one embodiment, in a lockstep mode and function as a single core with comparison logic to detect any differences between their operations. In at least one embodiment, processor(s) **1510** may further include a real-time camera engine that may include, without limitation, a dedicated processor subsystem for handling real-time camera management. In at least one embodiment, processor(s) **1510** may further include a high-dynamic range signal processor that may include, without limitation, an image signal processor that is a hardware engine that is part of camera processing pipeline.

[0161] In at least one embodiment, processor(s) **1510** may include a video image compositor that may be a processing block (e.g., implemented on a microprocessor) that implements video post-processing functions needed by a video playback application to produce final image for player window. In at least one embodiment, video image compositor may perform lens distortion correction on wide-view camera(s) **1570**, surround camera(s) **1574**, and/or on in-cabin monitoring camera sensor(s). In at least one embodiment, in-cabin monitoring camera sensor(s) are preferably monitored by a neural network running on another instance of SoC(s) **1504**, configured to identify in cabin events and respond accordingly. In at least one embodiment, an in-cabin system may perform, without limitation, lip reading to activate cellular service and place a phone call, dictate emails, change vehicle's destination, activate or change vehicle's infotainment system and settings, or provide voice-activated web surfing. In at least one embodiment, certain functions are available to driver when vehicle is operating in an autonomous mode and are disabled otherwise.

[0162] In at least one embodiment, video image compositor may include enhanced temporal noise reduction for both spatial and temporal noise reduction. For example, in at least one embodiment, where motion occurs in a video, noise reduction weights spatial information appropriately, decreasing weight of information provided by adjacent frames. In at least one embodiment, where an image or portion of an image does not include motion, temporal noise reduction performed by video image compositor may use information from previous image to reduce noise in current image.

[0163] In at least one embodiment, video image compositor may also be configured to perform stereo rectification on input stereo lens frames. In at least one embodiment, video image compositor may further be used for user interface composition when operating system desktop is in use, and GPU(s) **1508** are not required to continuously render new surfaces. In at least one embodiment, when GPU(s) **1508** are powered on and active doing 3D rendering, video image compositor may be used to offload GPU(s) **1508** to improve performance and responsiveness.

[0164] In at least one embodiment, one or more of SoC(s) **1504** may further include a mobile industry processor interface ("MIPI") camera serial interface for receiving video and input from cameras, a high-speed interface, and/or a video input block that may be used for camera and related pixel input functions. In at least one embodiment, one or more of SoC(s) **1504** may further include an input/output controller(s) that may be controlled by software and may be used for receiving I/O signals that are uncommitted to a specific role.

[0165] In at least one embodiment, one or more of SoC(s) **1504** may further include a broad range of peripheral interfaces to enable communication with peripherals, audio encoders/decoders ("codecs"), power management, and/or other devices. SoC(s) **1504** may be used to process data from cameras (e.g., connected over Gigabit Multimedia Serial Link and Ethernet), sensors (e.g., LIDAR sensor(s) **1564**,

RADAR sensor(s) **1560**, etc. that may be connected over Ethernet), data from bus **1502** (e.g., speed of vehicle **1500**, steering wheel position, etc.), data from GNSS sensor(s) **1558** (e.g., connected over Ethernet or CAN bus), etc. In at least one embodiment, one or more of SoC(s) **1504** may further include dedicated high-performance mass storage controllers that may include their own DMA engines, and that may be used to free CPU(s) **1506** from routine data management tasks.

[0166] In at least one embodiment, SoC(s) **1504** may be an end-to-end platform with a flexible architecture that spans automation levels 3-5, thereby providing a comprehensive functional safety architecture that leverages and makes efficient use of computer vision and ADAS techniques for diversity and redundancy, provides a platform for a flexible, reliable driving software stack, along with deep learning tools. In at least one embodiment, SoC(s) **1504** may be faster, more reliable, and even more energy-efficient and space-efficient than conventional systems. For example, in at least one embodiment, accelerator(s) **1514**, when combined with CPU(s) **1506**, GPU(s) **1508**, and data store(s) **1516**, may provide for a fast, efficient platform for level 3-5 autonomous vehicles.

[0167] In at least one embodiment, computer vision algorithms may be executed on CPUs, which may be configured using high-level programming language, such as C programming language, to execute a wide variety of processing algorithms across a wide variety of visual data. However, in at least one embodiment, CPUs are oftentimes unable to meet performance requirements of many computer vision applications, such as those related to execution time and power consumption, for example. In at least one embodiment, many CPUs are unable to execute complex object detection algorithms in real-time, which is used in in-vehicle ADAS applications and in practical Level 3-5 autonomous vehicles.

[0168] Embodiments described herein allow for multiple neural networks to be performed simultaneously and/or sequentially, and for results to be combined together to enable Level 3-5 autonomous driving functionality. For example, in at least one embodiment, a CNN executing on DLA or discrete GPU (e.g., GPU(s) **1520**) may include text and word recognition, allowing supercomputer to read and understand traffic signs, including signs for which neural network has not been specifically trained. In at least one embodiment, DLA may further include a neural network that is able to identify, interpret, and provide semantic understanding of sign, and to pass that semantic understanding to path planning modules running on CPU Complex.

[0169] In at least one embodiment, multiple neural networks may be run simultaneously, as for Level 3, 4, or 5 driving. For example, in at least one embodiment, a warning sign consisting of "Caution: flashing lights indicate icy conditions," along with an electric light, may be independently or collectively interpreted by several neural networks. In at least one embodiment, a sign itself may be identified as a traffic sign by a first deployed neural network (e.g., a neural network that has been trained) and a text "flashing lights indicate icy conditions" may be interpreted by a second deployed neural network, which informs vehicle's path planning software (preferably executing on CPU Complex) that when flashing lights are detected, icy conditions exist. In at least one embodiment, a flashing light may be identified by operating a third deployed neural network over multiple frames, informing vehicle's path-planning software of presence (or absence) of flashing lights. In at least one embodiment, all three neural networks may run simultaneously, such as within DLA and/or on GPU(s) **1508**.

[0170] In at least one embodiment, a CNN for facial recognition and vehicle owner identification may use data from camera sensors to identify presence of an authorized driver and/or owner of vehicle **1500**. In at least one embodiment, an always on sensor processing engine may be used to unlock vehicle when owner approaches driver door and turn on lights, and, in security mode, to disable vehicle when owner leaves vehicle. In this way, SoC(s) **1504** provide for security against theft and/or carjacking.

[0171] In at least one embodiment, a CNN for emergency vehicle detection and identification may use data from microphones **1596** to detect and identify emergency vehicle sirens. In at least one embodiment, SoC(s) **1504** use CNN for classifying environmental and urban sounds, as well as classifying visual data. In at least one embodiment, CNN running on DLA is trained to identify relative closing speed of emergency vehicle (e.g., by using Doppler effect). In at least one embodiment, CNN may also be trained to identify emergency vehicles specific to local area in which vehicle is operating, as identified by GNSS sensor(s) **1558**. In at least one embodiment, when operating in Europe, CNN will seek to detect European sirens, and when in United States CNN will seek to identify only North American sirens. In at least one embodiment, once an emergency vehicle is detected, a control program may be used to execute an emergency vehicle safety routine, slowing vehicle, pulling over to side of road, parking vehicle, and/or idling vehicle, with assistance of ultrasonic sensor(s) **1562**, until emergency vehicle (s) passes.

[0172] In at least one embodiment, vehicle **1500** may include CPU(s) **1518** (e.g., discrete CPU(s), or dCPU(s)), that may be coupled to SoC(s) **1504** via a high-speed interconnect (e.g., PCIe). In at least one embodiment, CPU (s) **1518** may include an X86 processor, for example. CPU(s) **1518** may be used to perform any of a variety of functions, including arbitrating potentially inconsistent results between ADAS sensors and SoC(s) **1504**, and/or monitoring status and health of controller(s) **1536** and/or an infotainment system on a chip ("infotainment SoC") **1530**, for example.

[0173] In at least one embodiment, vehicle **1500** may include GPU(s) **1520** (e.g., discrete GPU(s), or dGPU(s)), that may be coupled to SoC(s) **1504** via a high-speed interconnect (e.g., NVIDIA's NVLINK). In at least one embodiment, GPU(s) **1520** may provide additional artificial intelligence functionality, such as by executing redundant and/or different neural networks, and may be used to train and/or update neural networks based at least in part on input (e.g., sensor data) from sensors of vehicle **1500**.

[0174] In at least one embodiment, vehicle **1500** may further include network interface **1524** which may include, without limitation, wireless antenna(s) **1526** (e.g., one or more wireless antennas **1526** for different communication protocols, such as a cellular antenna, a Bluetooth antenna, etc.). In at least one embodiment, network interface **1524** may be used to enable wireless connectivity over Internet with cloud (e.g., with server(s) and/or other network devices), with other vehicles, and/or with computing devices (e.g., client devices of passengers). In at least one embodiment, to communicate with other vehicles, a direct link may

be established between vehicle **150** and other vehicle and/or an indirect link may be established (e.g., across networks and over Internet). In at least one embodiment, direct links may be provided using a vehicle-to-vehicle communication link. A vehicle-to-vehicle communication link may provide vehicle **1500** information about vehicles in proximity to vehicle **1500** (e.g., vehicles in front of, on side of, and/or behind vehicle **1500**). In at least one embodiment, aforementioned functionality may be part of a cooperative adaptive cruise control functionality of vehicle **1500**.

[0175] In at least one embodiment, network interface **1524** may include an SoC that provides modulation and demodulation functionality and enables controller(s) **1536** to communicate over wireless networks. In at least one embodiment, network interface **1524** may include a radio frequency front-end for up-conversion from baseband to radio frequency, and down conversion from radio frequency to baseband. In at least one embodiment, frequency conversions may be performed in any technically feasible fashion. For example, frequency conversions could be performed through well-known processes, and/or using super-heterodyne processes. In at least one embodiment, radio frequency front end functionality may be provided by a separate chip. In at least one embodiment, network interface may include wireless functionality for communicating over LTE, WCDMA, UMTS, GSM, CDMA2000, Bluetooth, Bluetooth LE, Wi-Fi, Z-Wave, ZigBee, LoRaWAN, and/or other wireless protocols.

[0176] In at least one embodiment, vehicle **1500** may further include data store(s) **1528** which may include, without limitation, off-chip (e.g., off SoC(s) **1504**) storage. In at least one embodiment, data store(s) **1528** may include, without limitation, one or more storage elements including RAM, SRAM, dynamic random-access memory ("DRAM"), video random-access memory ("VRAM"), Flash, hard disks, and/or other components and/or devices that may store at least one bit of data.

[0177] In at least one embodiment, vehicle **1500** may further include GNSS sensor(s) **1558** (e.g., GPS and/or assisted GPS sensors), to assist in mapping, perception, occupancy grid generation, and/or path planning functions. In at least one embodiment, any number of GNSS sensor(s) **1558** may be used, including, for example and without limitation, a GPS using a USB connector with an Ethernet to Serial (e.g., RS-232) bridge.

[0178] In at least one embodiment, vehicle **1500** may further include RADAR sensor(s) **1560**. RADAR sensor(s) **1560** may be used by vehicle **1500** for long-range vehicle detection, even in darkness and/or severe weather conditions. In at least one embodiment, RADAR functional safety levels may be ASIL B. RADAR sensor(s) **1560** may use CAN and/or bus **1502** (e.g., to transmit data generated by RADAR sensor(s) **1560**) for control and to access object tracking data, with access to Ethernet to access raw data in some examples. In at least one embodiment, wide variety of RADAR sensor types may be used. For example, and without limitation, RADAR sensor(s) **1560** may be suitable for front, rear, and side RADAR use. In at least one embodiment, one or more of RADAR sensors(s) **1560** are Pulse Doppler RADAR sensor(s).

[0179] In at least one embodiment, RADAR sensor(s) **1560** may include different configurations, such as long-range with narrow field of view, short-range with wide field of view, short-range side coverage, etc. In at least one embodiment, long-range RADAR may be used for adaptive cruise control functionality. In at least one embodiment, long-range RADAR systems may provide a broad field of view realized by two or more independent scans, such as within a 250m range. In at least one embodiment, RADAR sensor(s) **1560** may help in distinguishing between static and moving objects, and may be used by ADAS system **1538** for emergency brake assist and forward collision warning. Sensors **1560**(*s*) included in a long-range RADAR system may include, without limitation, monostatic multimodal RADAR with multiple (e.g., six or more) fixed RADAR antennae and a high-speed CAN and FlexRay interface. In at least one embodiment, with six antennae, central four antennae may create a focused beam pattern, designed to record vehicle **1500**'s surroundings at higher speeds with minimal interference from traffic in adjacent lanes. In at least one embodiment, other two antennae may expand field of view, making it possible to quickly detect vehicles entering or leaving vehicle **1500**'s lane.

[0180] In at least one embodiment, mid-range RADAR systems may include, as an example, a range of up to 160m (front) or 80m (rear), and a field of view of up to 42 degrees (front) or 150 degrees (rear). In at least one embodiment, short-range RADAR systems may include, without limitation, any number of RADAR sensor(s) **1560** designed to be installed at both ends of rear bumper. When installed at both ends of rear bumper, in at least one embodiment, a RADAR sensor system may create two beams that constantly monitor blind spot in rear and next to vehicle. In at least one embodiment, short-range RADAR systems may be used in ADAS system **1538** for blind spot detection and/or lane change assist.

[0181] In at least one embodiment, vehicle **1500** may further include ultrasonic sensor(s) **1562**. Ultrasonic sensor(s) **1562**, which may be positioned at front, back, and/or sides of vehicle **1500**, may be used for park assist and/or to create and update an occupancy grid. In at least one embodiment, a wide variety of ultrasonic sensor(s) **1562** may be used, and different ultrasonic sensor(s) **1562** may be used for different ranges of detection (e.g., 2.5m, 4m). In at least one embodiment, ultrasonic sensor(s) **1562** may operate at functional safety levels of ASIL B.

[0182] In at least one embodiment, vehicle **1500** may include LIDAR sensor(s) **1564**. LIDAR sensor(s) **1564** may be used for object and pedestrian detection, emergency braking, collision avoidance, and/or other functions. In at least one embodiment, LIDAR sensor(s) **1564** may be functional safety level ASIL B. In at least one embodiment, vehicle **1500** may include multiple LIDAR sensors **1564** (e.g., two, four, six, etc.) that may use Ethernet (e.g., to provide data to a Gigabit Ethernet switch).

[0183] In at least one embodiment, LIDAR sensor(s) **1564** may be capable of providing a list of objects and their distances for a 360-degree field of view. In at least one embodiment, commercially available LIDAR sensor(s) **1564** may have an advertised range of approximately 100m, with an accuracy of 2 cm-3 cm, and with support for a 100 Mbps Ethernet connection, for example. In at least one embodiment, one or more non-protruding LIDAR sensors **1564** may be used. In such an embodiment, LIDAR sensor(s) **1564** may be implemented as a small device that may be embedded into front, rear, sides, and/or corners of vehicle **1500**. In at least one embodiment, LIDAR sensor(s) **1564**, in such an embodiment, may provide up to a 120-degree

horizontal and 35-degree vertical field-of-view, with a 200m range even for low-reflectivity objects. In at least one embodiment, front-mounted LIDAR sensor(s) **1564** may be configured for a horizontal field of view between 45 degrees and 135 degrees.

[0184] In at least one embodiment, LIDAR technologies, such as 3D flash LIDAR, may also be used. 3D Flash LIDAR uses a flash of a laser as a transmission source, to illuminate surroundings of vehicle **1500** up to approximately 200m. In at least one embodiment, a flash LIDAR unit includes, without limitation, a receptor, which records laser pulse transit time and reflected light on each pixel, which in turn corresponds to range from vehicle **1500** to objects. In at least one embodiment, flash LIDAR may allow for highly accurate and distortion-free images of surroundings to be generated with every laser flash. In at least one embodiment, four flash LIDAR sensors may be deployed, one at each side of vehicle **1500**. In at least one embodiment, 3D flash LIDAR systems include, without limitation, a solid-state 3D staring array LIDAR camera with no moving parts other than a fan (e.g., a non-scanning LIDAR device). In at least one embodiment, flash LIDAR device(s) may use a 5 nanosecond class I (eye-safe) laser pulse per frame and may capture reflected laser light in form of 3D range point clouds and co-registered intensity data.

[0185] In at least one embodiment, vehicle may further include IMU sensor(s) **1566**. In at least one embodiment, IMU sensor(s) **1566** may be located at a center of rear axle of vehicle **1500**, in at least one embodiment. In at least one embodiment, IMU sensor(s) **1566** may include, for example and without limitation, accelerometer(s), magnetometer(s), gyroscope(s), magnetic compass(es), and/or other sensor types. In at least one embodiment, such as in six-axis applications, IMU sensor(s) **1566** may include, without limitation, accelerometers and gyroscopes. In at least one embodiment, such as in nine-axis applications, IMU sensor (s) **1566** may include, without limitation, accelerometers, gyroscopes, and magnetometers.

[0186] In at least one embodiment, IMU sensor(s) **1566** may be implemented as a miniature, high performance GPS-Aided Inertial Navigation System ("GPS/INS") that combines micro-electro-mechanical systems ("MEMS") inertial sensors, a high-sensitivity GPS receiver, and advanced Kalman filtering algorithms to provide estimates of position, velocity, and attitude. In at least one embodiment, IMU sensor(s) **1566** may enable vehicle **1500** to estimate heading without requiring input from a magnetic sensor by directly observing and correlating changes in velocity from GPS to IMU sensor(s) **1566**. In at least one embodiment, IMU sensor(s) **1566** and GNSS sensor(s) **1558** may be combined in a single integrated unit.

[0187] In at least one embodiment, vehicle **1500** may include microphone(s) **1596** placed in and/or around vehicle **1500**. In at least one embodiment, microphone(s) **1596** may be used for emergency vehicle detection and identification, among other things.

[0188] In at least one embodiment, vehicle **1500** may further include any number of camera types, including stereo camera(s) **1568**, wide-view camera(s) **1570**, infrared camera (s) **1572**, surround camera(s) **1574**, long-range camera(s) **1598**, mid-range camera(s) **1576**, and/or other camera types. In at least one embodiment, cameras may be used to capture image data around an entire periphery of vehicle **1500**. In at least one embodiment, types of cameras used depends on vehicle **1500**. In at least one embodiment, any combination of camera types may be used to provide necessary coverage around vehicle **1500**. In at least one embodiment, number of cameras may differ depending on embodiment. For example, in at least one embodiment, vehicle **1500** could include six cameras, seven cameras, ten cameras, twelve cameras, or another number of cameras. Cameras may support, as an example and without limitation, Gigabit Multimedia Serial Link ("GMSL") and/or Gigabit Ethernet. In at least one embodiment, each of camera(s) is described with more detail previously herein with respect to FIG. **15**A and FIG. **15**B.

[0189] In at least one embodiment, vehicle **1500** may further include vibration sensor(s) **1542**. In at least one embodiment, vibration sensor(s) **1542** may measure vibrations of components of vehicle **1500**, such as axle(s). For example, in at least one embodiment, changes in vibrations may indicate a change in road surfaces. In at least one embodiment, when two or more vibration sensors **1542** are used, differences between vibrations may be used to determine friction or slippage of road surface (e.g., when difference in vibration is between a power-driven axle and a freely rotating axle).

[0190] In at least one embodiment, vehicle **1500** may include ADAS system **1538**. ADAS system **1538** may include, without limitation, an SoC, in some examples. In at least one embodiment, ADAS system **1538** may include, without limitation, any number and combination of an autonomous/adaptive/automatic cruise control ("ACC") system, a cooperative adaptive cruise control ("CACC") system, a forward crash warning ("FCW") system, an automatic emergency braking ("AEB") system, a lane departure warning ("LDW)" system, a lane keep assist ("LKA") system, a blind spot warning ("BSW") system, a rear cross-traffic warning ("RCTW") system, a collision warning ("CW") system, a lane centering ("LC") system, and/or other systems, features, and/or functionality.

[0191] In at least one embodiment, ACC system may use RADAR sensor(s) **1560**, LIDAR sensor(s) **1564**, and/or any number of camera(s). In at least one embodiment, ACC system may include a longitudinal ACC system and/or a lateral ACC system. In at least one embodiment, longitudinal ACC system monitors and controls distance to vehicle immediately ahead of vehicle **1500** and automatically adjust speed of vehicle **1500** to maintain a safe distance from vehicles ahead. In at least one embodiment, lateral ACC system performs distance keeping, and advises vehicle **1500** to change lanes when necessary. In at least one embodiment, lateral ACC is related to other ADAS applications such as LC and CW.

[0192] In at least one embodiment, CACC system uses information from other vehicles that may be received via network interface **1524** and/or wireless antenna(s) **1526** from other vehicles via a wireless link, or indirectly, over a network connection (e.g., over Internet). In at least one embodiment, direct links may be provided by a vehicle-to-vehicle ("V2V") communication link, while indirect links may be provided by an infrastructure-to-vehicle ("I2V") communication link. In general, V2V communication concept provides information about immediately preceding vehicles (e.g., vehicles immediately ahead of and in same lane as vehicle **1500**), while I2V communication concept provides information about traffic further ahead. In at least one embodiment, CACC system may include either or both I2V and V2V information sources. In at least one embodi-

ment, given information of vehicles ahead of vehicle **1500**, CACC system may be more reliable and it has potential to improve traffic flow smoothness and reduce congestion on road.

[0193] In at least one embodiment, FCW system is designed to alert driver to a hazard, so that driver may take corrective action. In at least one embodiment, FCW system uses a front-facing camera and/or RADAR sensor(s) **1560**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component. In at least one embodiment, FCW system may provide a warning, such as in form of a sound, visual warning, vibration and/or a quick brake pulse.

[0194] In at least one embodiment, AEB system detects an impending forward collision with another vehicle or other object, and may automatically apply brakes if driver does not take corrective action within a specified time or distance parameter. In at least one embodiment, AEB system may use front-facing camera(s) and/or RADAR sensor(s) **1560**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC. In at least one embodiment, when AEB system detects a hazard, AEB system typically first alerts driver to take corrective action to avoid collision and, if driver does not take corrective action, AEB system may automatically apply brakes in an effort to prevent, or at least mitigate, impact of predicted collision. In at least one embodiment, AEB system, may include techniques such as dynamic brake support and/or crash imminent braking.

[0195] In at least one embodiment, LDW system provides visual, audible, and/or tactile warnings, such as steering wheel or seat vibrations, to alert driver when vehicle **1500** crosses lane markings. In at least one embodiment, LDW system does not activate when driver indicates an intentional lane departure, by activating a turn signal. In at least one embodiment, LDW system may use front-side facing cameras, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component. In at least one embodiment, LKA system is a variation of LDW system. LKA system provides steering input or braking to correct vehicle **1500** if vehicle **1500** starts to exit lane.

[0196] In at least one embodiment, BSW system detects and warns driver of vehicles in an automobile's blind spot. In at least one embodiment, BSW system may provide a visual, audible, and/or tactile alert to indicate that merging or changing lanes is unsafe. In at least one embodiment, BSW system may provide an additional warning when driver uses a turn signal. In at least one embodiment, BSW system may use rear-side facing camera(s) and/or RADAR sensor(s) **1560**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component.

[0197] In at least one embodiment, RCTW system may provide visual, audible, and/or tactile notification when an object is detected outside rear-camera range when vehicle **1500** is backing up. In at least one embodiment, RCTW system includes AEB system to ensure that vehicle brakes are applied to avoid a crash. In at least one embodiment, RCTW system may use one or more rear-facing RADAR sensor(s) **1560**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component.

[0198] In at least one embodiment, conventional ADAS systems may be prone to false positive results which may be annoying and distracting to a driver, but typically are not catastrophic, because conventional ADAS systems alert driver and allow driver to decide whether a safety condition truly exists and act accordingly. In at least one embodiment, vehicle **1500** itself decides, in case of conflicting results, whether to heed result from a primary computer or a secondary computer (e.g., first controller **1536** or second controller **1536**). For example, in at least one embodiment, ADAS system **1538** may be a backup and/or secondary computer for providing perception information to a backup computer rationality module. In at least one embodiment, backup computer rationality monitor may run a redundant diverse software on hardware components to detect faults in perception and dynamic driving tasks. In at least one embodiment, outputs from ADAS system **1538** may be provided to a supervisory MCU. In at least one embodiment, if outputs from primary computer and secondary computer conflict, supervisory MCU determines how to reconcile conflict to ensure safe operation.

[0199] In at least one embodiment, primary computer may be configured to provide supervisory MCU with a confidence score, indicating primary computer's confidence in chosen result. In at least one embodiment, if confidence score exceeds a threshold, supervisory MCU may follow primary computer's direction, regardless of whether secondary computer provides a conflicting or inconsistent result. In at least one embodiment, where confidence score does not meet threshold, and where primary and secondary computer indicate different results (e.g., a conflict), supervisory MCU may arbitrate between computers to determine appropriate outcome.

[0200] In at least one embodiment, supervisory MCU may be configured to run a neural network(s) that is trained and configured to determine, based at least in part on outputs from primary computer and secondary computer, conditions under which secondary computer provides false alarms. In at least one embodiment, neural network(s) in supervisory MCU may learn when secondary computer's output may be trusted, and when it cannot. For example, in at least one embodiment, when secondary computer is a RADAR-based FCW system, a neural network(s) in supervisory MCU may learn when FCW system is identifying metallic objects that are not, in fact, hazards, such as a drainage grate or manhole cover that triggers an alarm. In at least one embodiment, when secondary computer is a camera-based LDW system, a neural network in supervisory MCU may learn to override LDW when bicyclists or pedestrians are present and a lane departure is, in fact, safest maneuver. In at least one embodiment, supervisory MCU may include at least one of a DLA or GPU suitable for running neural network(s) with associated memory. In at least one embodiment, supervisory MCU may comprise and/or be included as a component of SoC(s) **1504**.

[0201] In at least one embodiment, ADAS system **1538** may include a secondary computer that performs ADAS functionality using traditional rules of computer vision. In at least one embodiment, secondary computer may use classic computer vision rules (if-then), and presence of a neural network(s) in supervisory MCU may improve reliability,

safety and performance. For example, in at least one embodiment, diverse implementation and intentional non-identity makes overall system more fault-tolerant, especially to faults caused by software (or software-hardware interface) functionality. For example, in at least one embodiment, if there is a software bug or error in software running on primary computer, and non-identical software code running on secondary computer provides same overall result, then supervisory MCU may have greater confidence that overall result is correct, and bug in software or hardware on primary computer is not causing material error.

[0202] In at least one embodiment, output of ADAS system **1538** may be fed into primary computer's perception block and/or primary computer's dynamic driving task block. For example, in at least one embodiment, if ADAS system **1538** indicates a forward crash warning due to an object immediately ahead, perception block may use this information when identifying objects. In at least one embodiment, secondary computer may have its own neural network which is trained and thus reduces risk of false positives, as described herein.

[0203] In at least one embodiment, vehicle **1500** may further include infotainment SoC **1530** (e.g., an in-vehicle infotainment system (IVI)). Although illustrated and described as an SoC, infotainment system **1530**, in at least one embodiment, may not be an SoC, and may include, without limitation, two or more discrete components. In at least one embodiment, infotainment SoC **1530** may include, without limitation, a combination of hardware and software that may be used to provide audio (e.g., music, a personal digital assistant, navigational instructions, news, radio, etc.), video (e.g., TV, movies, streaming, etc.), phone (e.g., hands-free calling), network connectivity (e.g., LTE, WiFi, etc.), and/or information services (e.g., navigation systems, rear-parking assistance, a radio data system, vehicle related information such as fuel level, total distance covered, brake fuel level, oil level, door open/close, air filter information, etc.) to vehicle **1500**. For example, infotainment SoC **1530** could include radios, disk players, navigation systems, video players, USB and Bluetooth connectivity, carputers, in-car entertainment, WiFi, steering wheel audio controls, hands free voice control, a heads-up display ("HUD"), HMI display **1534**, a telematics device, a control panel (e.g., for controlling and/or interacting with various components, features, and/or systems), and/or other components. In at least one embodiment, infotainment SoC **1530** may further be used to provide information (e.g., visual and/or audible) to user(s) of vehicle, such as information from ADAS system **1538**, autonomous driving information such as planned vehicle maneuvers, trajectories, surrounding environment information (e.g., intersection information, vehicle information, road information, etc.), and/or other information.

[0204] In at least one embodiment, infotainment SoC **1530** may include any amount and type of GPU functionality. In at least one embodiment, infotainment SoC **1530** may communicate over bus **1502** (e.g., CAN bus, Ethernet, etc.) with other devices, systems, and/or components of vehicle **1500**. In at least one embodiment, infotainment SoC **1530** may be coupled to a supervisory MCU such that GPU of infotainment system may perform some self-driving functions in event that primary controller(s) **1536** (e.g., primary and/or backup computers of vehicle **1500**) fail. In at least one embodiment, infotainment SoC **1530** may put vehicle **1500** into a chauffeur to safe stop mode, as described herein.

[0205] In at least one embodiment, vehicle **1500** may further include instrument cluster **1532** (e.g., a digital dash, an electronic instrument cluster, a digital instrument panel, etc.). In at least one embodiment, instrument cluster **1532** may include, without limitation, a controller and/or supercomputer (e.g., a discrete controller or supercomputer). In at least one embodiment, instrument cluster **1532** may include, without limitation, any number and combination of a set of instrumentation such as a speedometer, fuel level, oil pressure, tachometer, odometer, turn indicators, gearshift position indicator, seat belt warning light(s), parking-brake warning light(s), engine-malfunction light(s), supplemental restraint system (e.g., airbag) information, lighting controls, safety system controls, navigation information, etc. In some examples, information may be displayed and/or shared among infotainment SoC **1530** and instrument cluster **1532**. In at least one embodiment, instrument cluster **1532** may be included as part of infotainment SoC **1530**, or vice versa.

[0206] Inference and/or training logic **915** are used to perform inferencing and/or training operations associated with one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B. In at least one embodiment, inference and/or training logic **915** may be used in system FIG. **15**C 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.

[0207] Components of these figures can be used with layouts generated as discussed herein. In particular, these components can utilize road layouts for cities and other such regions for purposes such as navigation, testing, and training.

[0208] FIG. **15**D is a diagram of a system **1576** for communication between cloud-based server(s) and autonomous vehicle **1500** of FIG. **15**A, according to at least one embodiment. In at least one embodiment, system **1576** may include, without limitation, server(s) **1578**, network(s) **1590**, and any number and type of vehicles, including vehicle **1500**. In at least one embodiment, server(s) **1578** may include, without limitation, a plurality of GPUs **1584**(A)-**1584**(H) (collectively referred to herein as GPUs **1584**), PCIe switches **1582**(A)-**1582**(D) (collectively referred to herein as PCIe switches **1582**), and/or CPUs **1580**(A)-**1580**(B) (collectively referred to herein as CPUs **1580**). GPUs **1584**, CPUs **1580**, and PCIe switches **1582** may be interconnected with high-speed interconnects such as, for example and without limitation, NVLink interfaces **1588** developed by NVIDIA and/or PCIe connections **1586**. In at least one embodiment, GPUs **1584** are connected via an NVLink and/or NVSwitch SoC and GPUs **1584** and PCIe switches **1582** are connected via PCIe interconnects. In at least one embodiment, although eight GPUs **1584**, two CPUs **1580**, and four PCIe switches **1582** are illustrated, this is not intended to be limiting. In at least one embodiment, each of server(s) **1578** may include, without limitation, any number of GPUs **1584**, CPUs **1580**, and/or PCIe switches **1582**, in any combination. For example, in at least one embodiment, server(s) **1578** could each include eight, sixteen, thirty-two, and/or more GPUs **1584**.

[0209] In at least one embodiment, server(s) **1578** may receive, over network(s) **1590** and from vehicles, image data representative of images showing unexpected or changed road conditions, such as recently commenced road-work. In at least one embodiment, server(s) **1578** may transmit, over network(s) **1590** and to vehicles, neural networks **1592**, updated neural networks **1592**, and/or map information **1594**, including, without limitation, information regarding traffic and road conditions. In at least one embodiment, updates to map information **1594** may include, without limitation, updates for HD map **1522**, such as information regarding construction sites, potholes, detours, flooding, and/or other obstructions. In at least one embodiment, neural networks **1592**, updated neural networks **1592**, and/or map information **1594** may have resulted from new training and/or experiences represented in data received from any number of vehicles in environment, and/or based at least in part on training performed at a data center (e.g., using server(s) **1578** and/or other servers).

[0210] In at least one embodiment, server(s) **1578** may be used to train machine learning models (e.g., neural networks) based at least in part on training data. In at least one embodiment, training data may be generated by vehicles, and/or may be generated in a simulation (e.g., using a game engine). In at least one embodiment, any amount of training data is tagged (e.g., where associated neural network benefits from supervised learning) and/or undergoes other pre-processing. In at least one embodiment, any amount of training data is not tagged and/or pre-processed (e.g., where associated neural network does not require supervised learning). In at least one embodiment, once machine learning models are trained, machine learning models may be used by vehicles (e.g., transmitted to vehicles over network(s) **1590**, and/or machine learning models may be used by server(s) **1578** to remotely monitor vehicles.

[0211] In at least one embodiment, server(s) **1578** may receive data from vehicles and apply data to up-to-date real-time neural networks for real-time intelligent inferencing. In at least one embodiment, server(s) **1578** may include deep-learning supercomputers and/or dedicated AI computers powered by GPU(s) **1584**, such as a DGX and DGX Station machines developed by NVIDIA. However, in at least one embodiment, server(s) **1578** may include deep learning infrastructure that use CPU-powered data centers.

[0212] In at least one embodiment, deep-learning infrastructure of server(s) **1578** may be capable of fast, real-time inferencing, and may use that capability to evaluate and verify health of processors, software, and/or associated hardware in vehicle **1500**. For example, in at least one embodiment, deep-learning infrastructure may receive periodic updates from vehicle **1500**, such as a sequence of images and/or objects that vehicle **1500** has located in that sequence of images (e.g., via computer vision and/or other machine learning object classification techniques). In at least one embodiment, deep-learning infrastructure may run its own neural network to identify objects and compare them with objects identified by vehicle **1500** and, if results do not match and deep-learning infrastructure concludes that AI in vehicle **1500** is malfunctioning, then server(s) **1578** may transmit a signal to vehicle **1500** instructing a fail-safe computer of vehicle **1500** to assume control, notify passengers, and complete a safe parking maneuver.

[0213] In at least one embodiment, server(s) **1578** may include GPU(s) **1584** and one or more programmable inference accelerators (e.g., NVIDIA's TensorRT 3). In at least one embodiment, combination of GPU-powered servers and inference acceleration may make real-time responsiveness possible. In at least one embodiment, such as where performance is less critical, servers powered by CPUs, FPGAs, and other processors may be used for inferencing. In at least one embodiment, inference and/or training logic **915** are used to perform one or more embodiments. Details regarding inference and/or training logic **915** are provided below in conjunction with FIGS. **9**A and/or **9**B.

[0214] 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.

[0215] 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.

[0216] 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."

[0217] 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.

[0218] 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.

[0219] 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.

[0220] 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.

[0221] 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.

[0222] 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.

[0223] 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.

[0224] 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.

[0225] 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.

[0226] 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:
- receiving one or more initial regions of a spatial graph, the one or more initial regions including two or more nodes connected by one or more paths;
- identifying, for individual nodes of the spatial graph, one or more neighboring nodes within a determined proximity representing incoming paths to the individual nodes;
- predicting, using the one or more neighboring nodes and at least one determined style of the one or more initial regions, one or more additional nodes representing outgoing paths from the individual nodes; and
- iteratively predicting subsequent nodes from the one or more additional nodes until all nodes of the spatial graph have been analyzed or an end criterion is satisfied.

2. The computer-implemented method of claim **1**, further comprising:
- predicting the one or more additional nodes using a decoder recurrent neural network (RNN).

3. The computer-implemented method of claim **2**, further comprising:
- encoding local incoming paths from the neighboring nodes to the individual nodes using an encoder RNN, and providing data for the encoding as input to the decoder RNN.

4. The computer-implemented method of claim **1**, further comprising:
- adding the one or more additional nodes to a processing queue, the queue containing nodes that have not yet been analyzed for predicting the subsequent nodes.

5. The computer-implemented method of claim **1**, wherein the one or more initial regions are provided using at least one of a map, an aerial image, a sketch, or a description.

6. The computer-implemented method of claim **1**, further comprising:
- determining the at least one determined style of the one or more initial regions through receiving at least one style input or analyzing paths of the one or more initial regions to infer the at least one determined style.

7. The computer-implemented method of claim **1**, wherein predicting the one or more additional nodes is further based upon physical layout data represented in an aerial image.

8. The computer-implemented method of claim **1**, wherein the predicting is performed using a vector graphics-based generative neural network.

9. A system comprising:
- at least one processor; and
- memory including instructions that, when executed by the at least one processor, cause the system to:
  - determine a first node of a spatial layout;
  - determine a layout style to be used for expanding the spatial layout;
  - determine neighboring nodes within a determined distance of the first node and representing incoming paths to the first node; and
  - infer, using the layout style and information for the neighboring nodes, at least one second node of the spatial layout representing a new path segment from the first node in the spatial layout.

10. The system of claim **9**, wherein the instructions when executed further cause the system to:
- infer at least one second node for a plurality of first nodes of the spatial layout, the first node being any node of the spatial layout that has not yet been analyzed for a new path segment.

11. The system of claim **10**, wherein the instructions when executed further cause the system to:
- store each first node to a processing queue, wherein nodes are to be pulled from the processing queue to iteratively expand the spatial layout.

12. The system of claim **9**, wherein the instructions when executed further cause the system to:
- infer the at least one second node using a vector graphics-based generative neural network.

13. The system of claim **12**, wherein the vector graphics-based generative neural network includes an encoder-decoder architecture, wherein the encoder is an encoder RNN for encoding local incoming paths from the neighboring nodes to the first node, and wherein the decoder is a decoder RNN for inferring the at least one second node.

14. The system of claim **9**, wherein the instructions when executed further cause the system to:
- determine the layout style through receiving at least one style input or analyzing paths of the spatial layout.

15. The system of claim **9**, wherein the spatial layout corresponds to a road graph, wherein the first node corresponds to an intersection, and wherein the new path is a road segment leading from the intersection.

16. A road layout generator, comprising:
- an interface for receiving data indicative of at least one layout style;
- at least one processor; and
- memory including instructions that, when executed by the at least one processor, cause the road layout generator to:
  - determine pending nodes of a spatial graph that have not yet been analyzed by the road layout generator;
  - determine, for the pending nodes, neighboring nodes within a determined proximity of the pending nodes and representing incoming road segments to the pending nodes;
  - predict, using a neural network with the neighboring nodes and the at least one layout style, one or more additional nodes representing outgoing road segments from the pending nodes; and
  - iteratively predict subsequent nodes representing additional outgoing road segments for additional pending nodes of the spatial graph that have not yet been analyzed.

17. The road layout generator of claim **16**, wherein the instructions when executed further cause the system to:

predict the one or more additional nodes using a vector graphics-based generative neural network.

18. The road layout generator of claim **17**, wherein the vector graphics-based generative neural network includes an encoder-decoder architecture, wherein the encoder is an encoder RNN for encoding local incoming paths from the neighboring nodes to the pending nodes, and wherein the decoder is a decoder RNN for predicting the one or more additional nodes.

19. The road layout generator of claim **16**, wherein the data indicative of the layout style includes at least one of an aerial image of a location, a map, a sketch, a style specification, or an initial layout region.

20. The road layout generator of claim **16**, wherein the instructions when executed further cause the system to:

enable a user to provide layout information through an interactive interface, the one or more additional nodes being predicted further based on the provided layout information.

* * * * *