



(51) **International Patent Classification:**
G06N 3/0455 (2023.01) *G06N 3/082* (2023.01)

(21) **International Application Number:**
PCT/US2024/030384

(22) **International Filing Date:**
21 May 2024 (21.05.2024)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
63/504,563 26 May 2023 (26.05.2023) US
18/400,677 29 December 2023 (29.12.2023) US

(63) **Related by continuation (CON) or continuation-in-part (CIP) to earlier application:**

US 18/400,677 (CON)
Filed on 29 December 2023 (29.12.2023)

(71) **Applicant:** SNAP INC. [US/US]; 3000 31st Street, Santa Monica, California 90405 (US).

(72) **Inventors:** **CHEMERYS, Pavlo**; 3000 31st Street, Santa Monica, California 90405 (US). **ELES, Colin**; 3000 31st Street, Santa Monica, California 90405 (US). **HU, Ju**; 3000 31st Street, Santa Monica, California 90405 (US). **JIN, Qing**; 3000 31st Street, Santa Monica, California 90405 (US). **LI, Yanyu**; 3000 31st Street, Santa Monica, California 90405 (US). **MUCA, Ergeta**; 3000 31st Street, Santa Monica, California 90405 (US). **REN, Jian**; 3000 31st Street, Santa Monica, California 90405 (US). **SAGAR, Dhritiman**; 3000 31st Street, Santa Monica, California 90405 (US). **STOLIAR, Aleksei**; 3000 31st Street, Santa Monica, California 90405 (US). **TULYAKOV, Sergey**; 3000 31st Street, Santa Monica, California 90405 (US).

(54) **Title:** LATENT DIFFUSION MODEL AUTODECODERS

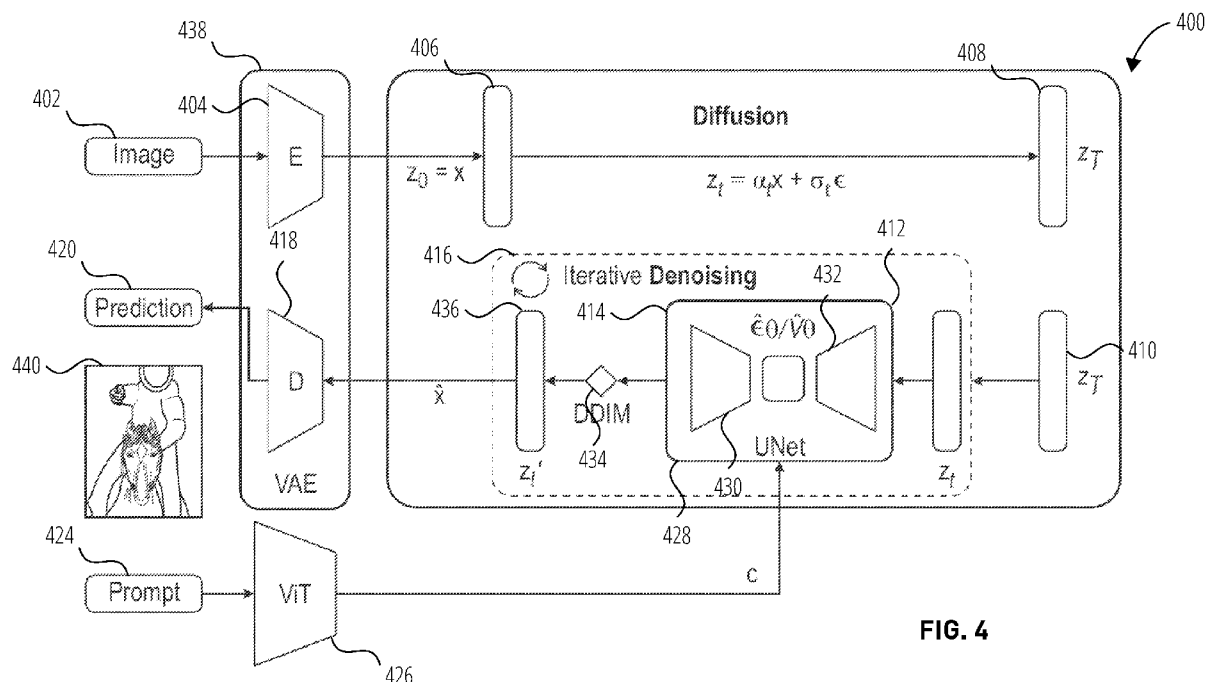


FIG. 4

(57) **Abstract:** Described is a system for improving machine learning models. In some cases, the system improves such models by identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model is trained to receive text as input and output an image based on the received text. The system identifies a number of channels in a decoder of the autoencoder, the decoder being configured to receive latent features as input and output images. The system further identifies a performance characteristic of the decoder and changes the node topology of the decoder based on the performance characteristic to generate an updated decoder. The system retrain the latent diffusion machine learning model using the updated decoder by inputting latent features to the updated decoder, receiving an outputted image from the updated decoder, and updating one or more weights of the decoder based on an assessment of the outputted image.

WANG, Huan; 3000 31st Street, Santa Monica, California 90405 (US).

(74) **Agent: PERDOK, Monique M.** et al.; Schwegman Lundberg & Woessner, P.A., P.O. Box 2938, Minneapolis, Minnesota 55402 (US).

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

(84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

LATENT DIFFUSION MODEL AUTODECODERS

CLAIM OF PRIORITY

[0001] This application claims the benefit of priority to U.S. Provisional Application Serial No. 63/504,563, filed on May 26, 2023, and U.S. Patent Application Serial No. 18/400,677, filed on December 29, 2023 each of which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates generally to machine learning models, and more specifically to text-to-image machine learning models.

BACKGROUND

[0003] As the popularity of Artificial Intelligence (AI) grows, companies use machine learning models in various ways, which is transforming how we process, analyze, and interact with visual data. The use of AI in image processing involves training algorithms, particularly deep learning models like Convolutional Neural Networks (CNNs), to perform tasks that range from low-level image manipulation to high-level understanding and generation of visual content. Some prominent applications of AI in images include image classification, object detection, image segmentation, facial recognition, and style transfer.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0004] In the drawings, which are not necessarily drawn to scale, like numerals may describe similar components in different views. To identify the discussion of any particular element or act more easily, the most significant digit or digits in a reference number refer to the figure number in which that element is first introduced. Some non-limiting examples are illustrated in the figures of the accompanying drawings in which:

[0005] FIG. 1 is a diagrammatic representation of a networked environment in which the present disclosure may be deployed, according to some examples.

[0006] FIG. 2 is a diagrammatic representation of an interaction system that has both client-side and server-side functionality, according to some examples.

[0007] FIG. 3 is a diagrammatic representation of a data structure as maintained in a database, according to some examples.

[0008] FIG. 4 illustrates an architecture for a latent diffusion model, according to some examples.

[0009] FIG. 5 illustrates an example routine for changing the node topology of a decoder for latent diffusion machine learning models, according to some examples.

[0010] FIG. 6 illustrates a latent diffusion machine learning model with an autoencoder to convert text to image, according to some examples.

[0011] FIG. 7 illustrates an updated decoder with removed channels and newly added channels, according to some examples.

[0012] FIG. 8 illustrates a comparison of the prior version of the decoder and the updated decoder for retraining, according to some examples.

5 [0013] FIG. 9 illustrates removing and adding channels efficiently to quickly identify an optimal decoder architecture, according to some examples.

[0014] FIG. 10 illustrates a modified decoder of a latent diffusion model, according to some examples.

10 [0015] FIG. 11 is a diagrammatic representation of a message, according to some examples.

[0016] FIG. 12 illustrates a system including a head-wearable apparatus with a selector input device, according to some examples.

[0017] FIG. 13 is a diagrammatic representation of a machine in the form of a computer system within which a set of instructions may be executed to cause the machine to perform
15 any one or more of the methodologies discussed herein, according to some examples.

[0018] FIG. 14 is a block diagram showing a software architecture within which examples may be implemented.

[0019] FIG. 15 illustrates a machine-learning pipeline, according to some examples.

20 [0020] FIG. 16 illustrates training and use of a machine-learning program, according to some examples.

DETAILED DESCRIPTION

[0021] Text-to-image machine learning models generate images based on textual descriptions. These models utilize deep learning techniques and are trained on large datasets of paired text and image examples. During the training phase, the model learns the
25 correlation between textual descriptions and corresponding images. Once trained, the model is used to generate images from textual descriptions. The generated image may not be an exact replica of the input description but captures the essence and key elements described in the text.

[0022] Text-to-image diffusion models create images from natural language descriptions
30 that rival the work of professional artists and photographers. However, these traditional models are large, with complex network architectures and require many denoising iterations, making them computationally expensive and slow to run. These models typically involve intricate network architectures and numerous denoising iterations, which increase computational complexity.

[0023] Another challenge is high computational costs. Running such models requires high-end Graphics Processing Units (GPUs) and often relies on cloud-based inference, restricting scalability and accessibility. Using cloud-based inference also involves sending user data to third-party servers, which raises privacy concerns as sensitive information is exposed to external entities. Users may be hesitant to share their data, especially when dealing with personal or confidential content. This approach is costly and has privacy implications, especially when user data is sent to a third party.

[0024] Diffusion-based text-to-image models show remarkable progress in synthesizing photo realistic content using text prompts. These models profoundly impact content creation, image editing and in-painting, super-resolution, video synthesis, 3D assets generation, and/or the like. However, this impact comes at the cost of the substantial increase in the computation requirements to run such models. As a result, to satisfy the necessary latency constraints, large scale cloud-based inference platforms with high-end GPU are required. This incurs high costs and brings potential privacy concerns, motivated by the sheer fact of sending private images, videos, and prompts to a third-party service.

[0025] There are emerging efforts attempting to speed the inference process of text-to-image diffusion models on mobile devices. Recent works use quantization or GPU-aware optimization to reduce the run time. While these methods effectively achieve a certain speed-up on mobile platforms, the latency does not allow for a seamless user experience. Moreover, none of the existing studies systematically examine the generation quality of on-device models through quantitative analysis.

[0026] Moreover, traditional diffusion models are not optimized for individual devices, such as mobile devices. Computational requirements and complex architectures make the models impractical to run on resource-constrained mobile platforms.

[0027] Furthermore, the extensive number of denoising steps in traditional models contributes to long inference times. This delay hampers real-time applications, interactive experiences, or scenarios where quick generation of images from textual descriptions is required or desired.

[0028] Due to the aforementioned technical challenges, traditional text-to-image diffusion models remain confined to a limited number of platforms with access to high-end hardware and computational resources. This restricts the widespread adoption of these powerful models.

[0029] To address these technical challenges, an example interaction system is described herein that applies various modifications to machine learning models that mitigate and/or eliminate the pitfalls of traditional latent diffusion models in the context of text-to-image

generation. Some variations involve retraining an existing machine learning model, whereas other variations involve architectural and functional changes to the machine learning models.

[0030] In some cases, the interaction system retraining a latent diffusion machine learning model that has been trained to perform iterative denoising. In some cases, the interaction system retraining a Variational Autoencoder (VAE) machine learning model by dynamically modifying a characteristic of an original decoder of the VAE machine learning model to generate a modified or updated decoder. The original decoder is initially trained to generate images from features in the latent space. The interaction system continuously modifies the original decoder by removing channels or layers based on a comparison between the output of the modified decoder with an original decoder. Advantageously, such data distillation pipelines compress and accelerate the decoder, while maintaining generative performance and reducing computational requirements.

[0031] By retraining and optimizing machine learning models, such as latent diffusion models and Variational Autoencoders (VAEs), the interaction system adapts the architecture, reduces channel redundancy, and fine-tunes the models. This results in accelerated inference, improved latency, reduced computational requirements, and maintained generative performance. These advancements make text-to-image generation accessible on various devices, including mobile platforms. By mitigating the limitations of traditional models, the interaction system paves the way for faster, more efficient, and higher quality text-to-image generation, empowering users to create visual content effortlessly.

[0032] In some cases, the interaction system reduces the required computation of the image decoder via data distillation. In some cases, interaction system applies a text-to-image diffusion model that generates an image on a mobile device in a very short amount of time (such as less than 2 seconds). These models improve the slow inference speed of the UNet (described further herein) and reduces the number of necessary denoising steps. The interaction system also introduces a data distillation pipeline to compress and accelerate the image decoder (as described further herein).

[0033] When the effects in this disclosure are considered in aggregate, one or more of the methodologies described herein may improve known systems, providing additional functionality (such as, but not limited to, the functionality mentioned above), making them easier, faster, or more intuitive to operate, and/or obviating a need for certain efforts or resources that otherwise would be involved in an image generation process. Computing

resources used by one or more machines, databases, or networks may thus be more efficiently utilized or even reduced.

[0034] NETWORKED COMPUTING ENVIRONMENT

[0035] FIG. 1 is a block diagram showing an example interaction system 100 for

5 facilitating interactions (e.g., exchanging text messages, conducting text audio and video calls, or playing games) over a network. The interaction system 100 includes multiple user systems 102, each of which hosts multiple applications, including an interaction client 104 and other applications 106. Each interaction client 104 is communicatively coupled, via one or more communication networks including a network 108 (e.g., the Internet), to other
10 instances of the interaction client 104 (e.g., hosted on respective other user systems 102), an interaction server system 110 and third-party servers 112). An interaction client 104 can also communicate with locally hosted applications 106 using Applications Programming Interfaces (APIs).

[0036] Each user system 102 may include multiple user devices, such as a mobile device
15 114, head-wearable apparatus 116, and a computer client device 118 that are communicatively connected to exchange data and messages.

[0037] An interaction client 104 interacts with other interaction clients 104 and with the interaction server system 110 via the network 108. The data exchanged between the interaction clients 104 (e.g., interactions 120) and between the interaction clients 104 and
20 the other interaction server system 110 includes functions (e.g., commands to invoke functions) and payload data (e.g., text, audio, video, or other multimedia data).

[0038] The interaction server system 110 provides server-side functionality via the network 108 to the interaction clients 104. While certain functions of the interaction system 100 are described herein as being performed by either an interaction client 104 or by the
25 interaction server system 110, the location of certain functionality either within the interaction client 104 or the interaction server system 110 may be a design choice. For example, it may be technically preferable to initially deploy particular technology and functionality within the interaction server system 110 but to later migrate this technology and functionality to the interaction client 104 where a user system 102 has sufficient
30 processing capacity.

[0039] The interaction server system 110 supports various services and operations that are provided to the interaction clients 104. Such operations include transmitting data to, receiving data from, and processing data generated by the interaction clients 104. This data may include message content, client device information, geolocation information, media
35 augmentation and overlays, message content persistence conditions, entity relationship

information, and live event information. Data exchanges within the interaction system 100 are invoked and controlled through functions available via user interfaces (UIs) of the interaction clients 104.

[0040] Turning now specifically to the interaction server system 110, an API server 122 is coupled to and provides programmatic interfaces to interaction servers 124, making the functions of the interaction servers 124 accessible to interaction clients 104, other applications 106 and third-party server 112. The interaction servers 124 are communicatively coupled to a database server 126, facilitating access to a database 128 that stores data associated with interactions processed by the interaction servers 124. Similarly, a web server 130 is coupled to the interaction servers 124 and provides web-based interfaces to the interaction servers 124. To this end, the web server 130 processes incoming network requests over the Hypertext Transfer Protocol (HTTP) and several other related protocols.

[0041] The API server 122 receives and transmits interaction data (e.g., commands and message payloads) between the interaction servers 124 and the user systems 102 (and, for example, interaction clients 104 and other application 106) and the third-party server 112. Specifically, the API server 122 provides a set of interfaces (e.g., routines and protocols) that can be called or queried by the interaction client 104 and other applications 106 to invoke functionality of the interaction servers 124. The API server 122 exposes various functions supported by the interaction servers 124, including account registration; login functionality; the sending of interaction data, via the interaction servers 124, from a particular interaction client 104 to another interaction client 104; the communication of media files (e.g., images or video) from an interaction client 104 to the interaction servers 124; the settings of a collection of media data (e.g., a story); the retrieval of a list of friends of a user of a user system 102; the retrieval of messages and content; the addition and deletion of entities (e.g., friends) to an entity relationship graph (e.g., the entity graph 310); the location of friends within an entity relationship graph; and opening an application event (e.g., relating to the interaction client 104).

[0042] The interaction servers 124 hosts multiple systems and subsystems, described below with reference to FIG. 2.

[0043] LINKED APPLICATIONS

[0044] Returning to the interaction client 104, features and functions of an external resource (e.g., a linked application 106 or applet) are made available to a user via an interface of the interaction client 104. In this context, "external" refers to the fact that the application 106 or applet is external to the interaction client 104. The external resource is often provided by a third party but may also be provided by the creator or provider of the

interaction client 104. The interaction client 104 receives a user selection of an option to launch or access features of such an external resource. The external resource may be the application 106 installed on the user system 102 (e.g., a "native app"), or a small-scale version of the application (e.g., an "applet") that is hosted on the user system 102 or remote of the user system 102 (e.g., on third-party servers 112). The small-scale version of the application includes a subset of features and functions of the application (e.g., the full-scale, native version of the application) and is implemented using a markup-language document. In some examples, the small-scale version of the application (e.g., an "applet") is a web-based, markup-language version of the application and is embedded in the interaction client 104. In addition to using markup-language documents (e.g., a *.ml file), an applet may incorporate a scripting language (e.g., a *.js file or a .json file) and a style sheet (e.g., a *.ss file).

[0045] In response to receiving a user selection of the option to launch or access features of the external resource, the interaction client 104 determines whether the selected external resource is a web-based external resource or a locally installed application 106. In some cases, applications 106 that are locally installed on the user system 102 can be launched independently of and separately from the interaction client 104, such as by selecting an icon corresponding to the application 106 on a home screen of the user system 102. Small-scale versions of such applications can be launched or accessed via the interaction client 104 and, in some examples, no or limited portions of the small-scale application can be accessed outside of the interaction client 104. The small-scale application can be launched by the interaction client 104 receiving, from third-party servers 112 for example, a markup-language document associated with the small-scale application and processing such a document.

[0046] In response to determining that the external resource is a locally installed application 106, the interaction client 104 instructs the user system 102 to launch the external resource by executing locally stored code corresponding to the external resource. In response to determining that the external resource is a web-based resource, the interaction client 104 communicates with the third-party servers 112 (for example) to obtain a markup-language document corresponding to the selected external resource. The interaction client 104 then processes the obtained markup-language document to present the web-based external resource within a user interface of the interaction client 104.

[0047] The interaction client 104 can notify a user of the user system 102, or other users related to such a user (e.g., "friends"), of activity taking place in one or more external resources. For example, the interaction client 104 can provide participants in a conversation

(e.g., a chat session) in the interaction client 104 with notifications relating to the current or recent use of an external resource by one or more members of a group of users. One or more users can be invited to join in an active external resource or to launch a recently used but currently inactive (in the group of friends) external resource. The external resource can provide participants in a conversation, each using respective interaction clients 104, with the ability to share an item, status, state, or location in an external resource in a chat session with one or more members of a group of users. The shared item may be an interactive chat card with which members of the chat can interact, for example, to launch the corresponding external resource, view specific information within the external resource, or take the member of the chat to a specific location or state within the external resource. Within a given external resource, response messages can be sent to users on the interaction client 104. The external resource can selectively include different media items in the responses, based on a current context of the external resource.

[0048] The interaction client 104 can present a list of the available external resources (e.g., applications 106 or applets) to a user to launch or access a given external resource. This list can be presented in a context-sensitive menu. For example, the icons representing different applications 106 (or applets) can vary based on how the menu is launched by the user (e.g., from a conversation interface or from a non-conversation interface).

[0049] SYSTEM ARCHITECTURE

[0050] FIG. 2 is a block diagram illustrating further details regarding the interaction system 100, according to some examples. Specifically, the interaction system 100 is shown to comprise the interaction client 104 and the interaction servers 124. The interaction system 100 embodies multiple subsystems, which are supported on the client-side by the interaction client 104 and on the server-side by the interaction servers 124. In some examples, these subsystems are implemented as microservices. A microservice subsystem (e.g., a microservice application) may have components that enable it to operate independently and communicate with other services. Example components of a microservice subsystem may include:

- **Function logic:** The function logic implements the functionality of the microservice subsystem, representing a specific capability or function that the microservice provides.
- **API interface:** Microservices may communicate with other component through well-defined APIs or interfaces, using lightweight protocols such as REST or messaging. The API interface defines the inputs and outputs of the microservice subsystem and how it interacts with other microservice subsystems of the interaction system 100.

- Data storage: A microservice subsystem may be responsible for its own data storage, which may be in the form of a database, cache, or other storage mechanism (e.g., using the database server 126 and database 128). This enables a microservice subsystem to operate independently of other microservices of the interaction system 100.

5 • Service discovery: Microservice subsystems may find and communicate with other microservice subsystems of the interaction system 100. Service discovery mechanisms enable microservice subsystems to locate and communicate with other microservice subsystems in a scalable and efficient way.

10 • Monitoring and logging: Microservice subsystems may need to be monitored and logged in order to ensure availability and performance. Monitoring and logging mechanisms enable the tracking of health and performance of a microservice subsystem.

[0051] In some examples, the interaction system 100 may employ a monolithic architecture, a service-oriented architecture (SOA), a function-as-a-service (FaaS) architecture, or a modular architecture:

15 [0052] Example subsystems are discussed below.

[0053] An image processing system 202 provides various functions that enable a user to capture and augment (e.g., annotate or otherwise modify or edit) media content associated with a message.

20 [0054] A camera system 204 includes control software (e.g., in a camera application) that interacts with and controls camera hardware (e.g., directly or via operating system controls) of the user system 102 to modify and augment real-time images captured and displayed via the interaction client 104.

25 [0055] The augmentation system 206 provides functions related to the generation and publishing of augmentations (e.g., media overlays) for images captured in real-time by cameras of the user system 102 or retrieved from memory of the user system 102. For example, the augmentation system 206 operatively selects, presents, and displays media overlays (e.g., an image filter or an image lens) to the interaction client 104 for the augmentation of real-time images received via the camera system 204 or stored images retrieved from memory 1202 of a user system 102. These augmentations are selected by the
30 augmentation system 206 and presented to a user of an interaction client 104, based on a number of inputs and data, such as for example:

- Geolocation of the user system 102; and
- Entity relationship information of the user of the user system 102.

[0056] An augmentation may include audio and visual content and visual effects.

35 Examples of audio and visual content include pictures, texts, logos, animations, and sound

effects. An example of a visual effect includes color overlaying. The audio and visual content or the visual effects can be applied to a media content item (e.g., a photo or video) at user system 102 for communication in a message, or applied to video content, such as a video content stream or feed transmitted from an interaction client 104. As such, the image processing system 202 may interact with, and support, the various subsystems of the communication system 208, such as the messaging system 210 and the video communication system 212.

[0057] A media overlay may include text or image data that can be overlaid on top of a photograph taken by the user system 102 or a video stream produced by the user system 102. In some examples, the media overlay may be a location overlay (e.g., Venice beach), a name of a live event, or a name of a merchant overlay (e.g., Beach Coffee House). In further examples, the image processing system 202 uses the geolocation of the user system 102 to identify a media overlay that includes the name of a merchant at the geolocation of the user system 102. The media overlay may include other indicia associated with the merchant. The media overlays may be stored in the databases 128 and accessed through the database server 126.

[0058] The image processing system 202 provides a user-based publication platform that enables users to select a geolocation on a map and upload content associated with the selected geolocation. The user may also specify circumstances under which a particular media overlay should be offered to other users. The image processing system 202 generates a media overlay that includes the uploaded content and associates the uploaded content with the selected geolocation.

[0059] The augmentation creation system 214 supports augmented reality developer platforms and includes an application for content creators (e.g., artists and developers) to create and publish augmentations (e.g., augmented reality experiences) of the interaction client 104. The augmentation creation system 214 provides a library of built-in features and tools to content creators including, for example custom shaders, tracking technology, and templates.

[0060] In some examples, the augmentation creation system 214 provides a merchant-based publication platform that enables merchants to select a particular augmentation associated with a geolocation via a bidding process. For example, the augmentation creation system 214 associates a media overlay of the highest bidding merchant with a corresponding geolocation for a predefined amount of time.

[0061] A communication system 208 is responsible for enabling and processing multiple forms of communication and interaction within the interaction system 100 and includes a

messaging system 210, an audio communication system 216, and a video communication system 212. The messaging system 210 is responsible for enforcing the temporary or time-limited access to content by the interaction clients 104. The messaging system 210 incorporates multiple timers (e.g., within an ephemeral timer system) that, based on duration and display parameters associated with a message or collection of messages (e.g., a story), selectively enable access (e.g., for presentation and display) to messages and associated content via the interaction client 104. The audio communication system 216 enables and supports audio communications (e.g., real-time audio chat) between multiple interaction clients 104. Similarly, the video communication system 212 enables and supports video communications (e.g., real-time video chat) between multiple interaction clients 104.

[0062] A user management system 218 is operationally responsible for the management of user data and profiles, and maintains entity information (e.g., stored in entity tables 308, entity graphs 310 and profile data 302) regarding users and relationships between users of the interaction system 100.

[0063] A collection management system 220 is operationally responsible for managing sets or collections of media (e.g., collections of text, image video, and audio data). A collection of content (e.g., messages, including images, video, text, and audio) may be organized into an "event gallery" or an "event story." Such a collection may be made available for a specified time period, such as the duration of an event to which the content relates. For example, content relating to a music concert may be made available as a "story" for the duration of that music concert. The collection management system 220 may also be responsible for publishing an icon that provides notification of a particular collection to the user interface of the interaction client 104. The collection management system 220 includes a curation function that allows a collection manager to manage and curate a particular collection of content. For example, the curation interface enables an event organizer to curate a collection of content relating to a specific event (e.g., delete inappropriate content or redundant messages). Additionally, the collection management system 220 employs machine vision (or image recognition technology) and content rules to curate a content collection automatically. In certain examples, compensation may be paid to a user to include user-generated content into a collection. In such cases, the collection management system 220 operates to automatically make payments to such users to use their content.

[0064] A map system 222 provides various geographic location (e.g., geolocation) functions and supports the presentation of map-based media content and messages by the interaction client 104. For example, the map system 222 enables the display of user icons or

avatars (e.g., stored in profile data 302) on a map to indicate a current or past location of "friends" of a user, as well as media content (e.g., collections of messages including photographs and videos) generated by such friends, within the context of a map. For example, a message posted by a user to the interaction system 100 from a specific

5 geographic location may be displayed within the context of a map at that particular location to "friends" of a specific user on a map interface of the interaction client 104. A user can furthermore share his or her location and status information (e.g., using an appropriate status avatar) with other users of the interaction system 100 via the interaction client 104, with this location and status information being similarly displayed within the context of a
10 map interface of the interaction client 104 to selected users.

[0065] A game system 224 provides various gaming functions within the context of the interaction client 104. The interaction client 104 provides a game interface providing a list of available games that can be launched by a user within the context of the interaction client 104 and played with other users of the interaction system 100. The interaction system 100
15 further enables a particular user to invite other users to participate in the play of a specific game by issuing invitations to such other users from the interaction client 104. The interaction client 104 also supports audio, video, and text messaging (e.g., chats) within the context of gameplay, provides a leaderboard for the games, and also supports the provision of in-game rewards (e.g., coins and items).

[0066] An external resource system 226 provides an interface for the interaction client 104 to communicate with remote servers (e.g., third-party servers 112) to launch or access external resources, i.e., applications or applets. Each third-party server 112 hosts, for example, a markup language (e.g., HTML5) based application or a small-scale version of an application (e.g., game, utility, payment, or ride-sharing application). The interaction client
25 104 may launch a web-based resource (e.g., application) by accessing the HTML5 file from the third-party servers 112 associated with the web-based resource. Applications hosted by third-party servers 112 are programmed in JavaScript leveraging a Software Development Kit (SDK) provided by the interaction servers 124. The SDK includes APIs with functions that can be called or invoked by the web-based application. The interaction servers 124
30 hosts a JavaScript library that provides a given external resource access to specific user data of the interaction client 104. HTML5 is an example of technology for programming games, but applications and resources programmed based on other technologies can be used.

[0067] To integrate the functions of the SDK into the web-based resource, the SDK is downloaded by the third-party server 112 from the interaction servers 124 or is otherwise
35 received by the third-party server 112. Once downloaded or received, the SDK is included

as part of the application code of a web-based external resource. The code of the web-based resource can then call or invoke certain functions of the SDK to integrate features of the interaction client 104 into the web-based resource.

[0068] The SDK stored on the interaction server system 110 effectively provides the bridge between an external resource (e.g., applications 106 or applets) and the interaction client 104. This gives the user a seamless experience of communicating with other users on the interaction client 104 while also preserving the look and feel of the interaction client 104. To bridge communications between an external resource and an interaction client 104, the SDK facilitates communication between third-party servers 112 and the interaction client 104. A bridge script running on a user system 102 establishes two one-way communication channels between an external resource and the interaction client 104. Messages are sent between the external resource and the interaction client 104 via these communication channels asynchronously. Each SDK function invocation is sent as a message and callback. Each SDK function is implemented by constructing a unique callback identifier and sending a message with that callback identifier.

[0069] By using the SDK, not all information from the interaction client 104 is shared with third-party servers 112. The SDK limits which information is shared based on the needs of the external resource. Each third-party server 112 provides an HTML5 file corresponding to the web-based external resource to interaction servers 124. The interaction servers 124 can add a visual representation (such as a box art or other graphic) of the web-based external resource in the interaction client 104. Once the user selects the visual representation or instructs the interaction client 104 through a graphical user interface (GUI) of the interaction client 104 to access features of the web-based external resource, the interaction client 104 obtains the HTML5 file and instantiates the resources to access the features of the web-based external resource.

[0070] The interaction client 104 presents a graphical user interface (e.g., a landing page or title screen) for an external resource. During, before, or after presenting the landing page or title screen, the interaction client 104 determines whether the launched external resource has been previously authorized to access user data of the interaction client 104. In response to determining that the launched external resource has been previously authorized to access user data of the interaction client 104, the interaction client 104 presents another graphical user interface of the external resource that includes functions and features of the external resource. In response to determining that the launched external resource has not been previously authorized to access user data of the interaction client 104, after a threshold period of time (e.g., 3 seconds) of displaying the landing page or title screen of the external

resource, the interaction client 104 slides up (e.g., animates a menu as surfacing from a bottom of the screen to a middle or other portion of the screen) a menu for authorizing the external resource to access the user data. The menu identifies the type of user data that the external resource will be authorized to use. In response to receiving a user selection of an accept option, the interaction client 104 adds the external resource to a list of authorized external resources and allows the external resource to access user data from the interaction client 104. The external resource is authorized by the interaction client 104 to access the user data under an OAuth 2 framework.

[0071] The interaction client 104 controls the type of user data that is shared with external resources based on the type of external resource being authorized. For example, external resources that include full-scale applications (e.g., an application 106) are provided with access to a first type of user data (e.g., two-dimensional avatars of users with or without different avatar characteristics). As another example, external resources that include small-scale versions of applications (e.g., web-based versions of applications) are provided with access to a second type of user data (e.g., payment information, two-dimensional avatars of users, three-dimensional avatars of users, and avatars with various avatar characteristics). Avatar characteristics include different ways to customize a look and feel of an avatar, such as different poses, facial features, clothing, and so forth.

[0072] An advertisement system 228 operationally enables the purchasing of advertisements by third parties for presentation to end-users via the interaction clients 104 and also handles the delivery and presentation of these advertisements.

[0073] An artificial intelligence and machine learning system 230 provides a variety of services to different subsystems within the interaction system 100. For example, the artificial intelligence and machine learning system 230 operates with the image processing system 202 and the camera system 204 to analyze images and extract information such as objects, text, or faces. This information can then be used by the image processing system 202 to enhance, filter, or manipulate images. The artificial intelligence and machine learning system 230 may be used by the augmentation system 206 to generate augmented content and augmented reality experiences, such as adding virtual objects or animations to real-world images. The communication system 208 and messaging system 210 may use the artificial intelligence and machine learning system 230 to analyze communication patterns and provide insights into how users interact with each other and provide intelligent message classification and tagging, such as categorizing messages based on sentiment or topic. The artificial intelligence and machine learning system 230 may also provide chatbot functionality to message interactions 120 between user systems 102 and between a user

system 102 and the interaction server system 110. The artificial intelligence and machine learning system 230 may also work with the audio communication system 216 to provide speech recognition and natural language processing capabilities, allowing users to interact with the interaction system 100 using voice commands.

5 [0074] DATA ARCHITECTURE

[0075] FIG. 3 is a schematic diagram illustrating data structures 300, which may be stored in the database 304 of the interaction server system 110, according to certain examples.

While the content of the database 304 is shown to comprise multiple tables, it will be appreciated that the data could be stored in other types of data structures (e.g., as an object-oriented database).

[0076] The database 304 includes message data stored within a message table 306. This message data includes, for any particular message, at least message sender data, message recipient (or receiver) data, and a payload. Further details regarding information that may be included in a message and included within the message data stored in the message table 306, are described below with reference to FIG. 3.

[0077] An entity table 308 stores entity data, and is linked (e.g., referentially) to an entity graph 310 and profile data 302. Entities for which records are maintained within the entity table 308 may include individuals, corporate entities, organizations, objects, places, events, and so forth. Regardless of entity type, any entity regarding which the interaction server system 110 stores data may be a recognized entity. Each entity is provided with a unique identifier, as well as an entity type identifier (not shown).

[0078] The entity graph 310 stores information regarding relationships and associations between entities. Such relationships may be social, professional (e.g., work at a common corporation or organization), interest-based, or activity-based, merely for example. Certain relationships between entities may be unidirectional, such as a subscription by an individual user to digital content of a commercial or publishing user (e.g., a newspaper or other digital media outlet, or a brand). Other relationships may be bidirectional, such as a "friend" relationship between individual users of the interaction system 100. A friend relationship can be established by mutual agreement between two entities. This mutual agreement may be established by an offer from a first entity to a second entity to establish a friend relationship, and acceptance by the second entity of the offer for establishment of the friend relationship.

[0079] The profile data 302 stores multiple types of profile data about a particular entity. The profile data 302 may be selectively used and presented to other users of the interaction system 100 based on privacy settings specified by a particular entity. Where the entity is an

individual, the profile data 302 includes, for example, a user name, telephone number, address, settings (e.g., notification and privacy settings), as well as a user-selected avatar representation (or collection of such avatar representations). A particular user may then selectively include one or more of these avatar representations within the content of messages communicated via the interaction system 100, and on map interfaces displayed by interaction clients 104 to other users. The collection of avatar representations may include "status avatars," which present a graphical representation of a status or activity that the user may select to communicate at a particular time.

[0080] Where the entity is a group, the profile data 302 for the group may similarly include one or more avatar representations associated with the group, in addition to the group name, members, and various settings (e.g., notifications) for the relevant group.

[0081] The database 304 also stores augmentation data, such as overlays or filters, in an augmentation table 312. The augmentation data is associated with and applied to videos (for which data is stored in a video table 314) and images (for which data is stored in an image table 316).

[0082] Filters, in some examples, are overlays that are displayed as overlaid on an image or video during presentation to a recipient user. Filters may be of various types, including user-selected filters from a set of filters presented to a sending user by the interaction client 104 when the sending user is composing a message. Other types of filters include geolocation filters (also known as geo-filters), which may be presented to a sending user based on geographic location. For example, geolocation filters specific to a neighborhood or special location may be presented within a user interface by the interaction client 104, based on geolocation information determined by a Global Positioning System (GPS) unit of the user system 102.

[0083] Another type of filter is a data filter, which may be selectively presented to a sending user by the interaction client 104 based on other inputs or information gathered by the user system 102 during the message creation process. Examples of data filters include current temperature at a specific location, a current speed at which a sending user is traveling, battery life for a user system 102, or the current time.

[0084] Other augmentation data that may be stored within the image table 316 includes augmented reality content items (e.g., corresponding to applying "lenses" or augmented reality experiences). An augmented reality content item may be a real-time special effect and sound that may be added to an image or a video.

[0085] As described above, augmentation data includes augmented reality content items, overlays, image transformations, AR images, and similar terms refer to modifications that

may be applied to image data (e.g., videos or images). This includes real-time modifications, which modify an image as it is captured using device sensors (e.g., one or multiple cameras) of the user system 102 and then displayed on a screen of the user system 102 with the modifications. This also includes modifications to stored content, such as video clips in a collection or group that may be modified. For example, in a user system 102 with access to multiple augmented reality content items, a user can use a single video clip with multiple augmented reality content items to see how the different augmented reality content items will modify the stored clip. Similarly, real-time video capture may use modifications to show how video images currently being captured by sensors of a user system 102 would modify the captured data. Such data may simply be displayed on the screen and not stored in memory, or the content captured by the device sensors may be recorded and stored in memory with or without the modifications (or both). In some systems, a preview feature can show how different augmented reality content items will look within different windows in a display at the same time. This can, for example, enable multiple windows with different pseudo random animations to be viewed on a display at the same time.

[0086] Data and various systems using augmented reality content items or other such transform systems to modify content using this data can thus involve detection of objects (e.g., faces, hands, bodies, cats, dogs, surfaces, objects, etc.), tracking of such objects as they leave, enter, and move around the field of view in video frames, and the modification or transformation of such objects as they are tracked. In various examples, different methods for achieving such transformations may be used. Some examples may involve generating a three-dimensional mesh model of the object or objects and using transformations and animated textures of the model within the video to achieve the transformation. In some examples, tracking of points on an object may be used to place an image or texture (which may be two-dimensional or three-dimensional) at the tracked position. In still further examples, neural network analysis of video frames may be used to place images, models, or textures in content (e.g., images or frames of video). Augmented reality content items thus refer both to the images, models, and textures used to create transformations in content, as well as to additional modeling and analysis information needed to achieve such transformations with object detection, tracking, and placement.

[0087] Real-time video processing can be performed with any kind of video data (e.g., video streams, video files, etc.) saved in a memory of a computerized system of any kind. For example, a user can load video files and save them in a memory of a device or can generate a video stream using sensors of the device. Additionally, any objects can be

processed using a computer animation model, such as a human's face and parts of a human body, animals, or non-living things such as chairs, cars, or other objects.

[0088] In some examples, when a particular modification is selected along with content to be transformed, elements to be transformed are identified by the computing device, and then detected and tracked if they are present in the frames of the video. The elements of the object are modified according to the request for modification, thus transforming the frames of the video stream. Transformation of frames of a video stream can be performed by different methods for different kinds of transformation. For example, for transformations of frames mostly referring to changing forms of object's elements characteristic points for each element of an object are calculated. Then, a mesh based on the characteristic points is generated for each element of the object. This mesh is used in the following stage of tracking the elements of the object in the video stream. In the process of tracking, the mesh for each element is aligned with a position of each element. Then, additional points are generated on the mesh.

[0089] In some examples, transformations changing some areas of an object using its elements can be performed by calculating characteristic points for each element of an object and generating a mesh based on the calculated characteristic points. Points are generated on the mesh, and then various areas based on the points are generated. The elements of the object are then tracked by aligning the area for each element with a position for each of the at least one element, and properties of the areas can be modified based on the request for modification, thus transforming the frames of the video stream. Depending on the specific request for modification properties of the mentioned areas can be transformed in different ways. Such modifications may involve changing the color of areas; removing some part of areas from the frames of the video stream; including new objects into areas that are based on a request for modification; and modifying or distorting the elements of an area or object. In various examples, any combination of such modifications or other similar modifications may be used. For certain models to be animated, some characteristic points can be selected as control points to be used in determining the entire state-space of options for the model animation. In some examples of a computer animation model to transform image data using face detection, the face is detected on an image using a specific face detection algorithm (e.g., Viola-Jones). Then, an Active Shape Model (ASM) algorithm is applied to the face region of an image to detect facial feature reference points.

[0090] Other methods and algorithms suitable for face detection can be used. For example, in some examples, features are located using a landmark, which represents a distinguishable point present in most of the images under consideration. For facial landmarks, for example,

the location of the left eye pupil may be used. If an initial landmark is not identifiable (e.g., if a person has an eyepatch), secondary landmarks may be used. Such landmark identification procedures may be used for any such objects. In some examples, a set of landmarks forms a shape. Shapes can be represented as vectors using the coordinates of the points in the shape. One shape is aligned to another with a similarity transform (allowing translation, scaling, and rotation) that minimizes the average Euclidean distance between shape points. The mean shape is the mean of the aligned training shapes.

[0091] The system can capture an image or video stream on a client device (e.g., the user system 102) and perform complex image manipulations locally on the user system 102 while maintaining a suitable user experience, computation time, and power consumption. The complex image manipulations may include size and shape changes, emotion transfers (e.g., changing a face from a frown to a smile), state transfers (e.g., aging a subject, reducing apparent age, changing gender), style transfers, graphical element application, and any other suitable image or video manipulation implemented by a convolutional neural network that has been configured to execute efficiently on the user system 102.

[0092] In some examples, the system operating within the interaction client 104 determines the presence of a face within the image or video stream and provides modification icons associated with a computer animation model to transform image data, or the computer animation model can be present as associated with an interface described herein. The system may implement a complex convolutional neural network on a portion of the image or video stream to generate and apply the selected modification. That is, the user may capture the image or video stream and be presented with a modified result in real-time or near real-time once a modification icon has been selected. Further, the modification may be persistent while the video stream is being captured, and the selected modification icon remains toggled. Machine-taught neural networks may be used to enable such modifications.

[0093] A collections table 318 stores data regarding collections of messages and associated image, video, or audio data, which are compiled into a collection (e.g., a story or a gallery). The creation of a particular collection may be initiated by a particular user (e.g., each user for which a record is maintained in the entity table 308). A user may create a "personal story" in the form of a collection of content that has been created and sent/broadcast by that user. To this end, the user interface of the interaction client 104 may include an icon that is user-selectable to enable a sending user to add specific content to his or her personal story.

[0094] A collection may also constitute a "live story," which is a collection of content from multiple users that is created manually, automatically, or using a combination of

manual and automatic techniques. For example, a "live story" may constitute a curated stream of user-submitted content from various locations and events. Users whose client devices have location services enabled and are at a common location event at a particular time may, for example, be presented with an option, via a user interface of the interaction client 104, to contribute content to a particular live story. The live story may be identified to the user by the interaction client 104, based on his or her location. The end result is a "live story" told from a community perspective.

[0095] A further type of content collection is known as a "location story," which enables a user whose user system 102 is located within a specific geographic location (e.g., on a college or university campus) to contribute to a particular collection. In some examples, a contribution to a location story may employ a second degree of authentication to verify that the end-user belongs to a specific organization or other entity (e.g., is a student on the university campus).

[0096] As mentioned above, the video table 314 stores video data that, in some examples, is associated with messages for which records are maintained within the message table 306. Similarly, the image table 316 stores image data associated with messages for which message data is stored in the entity table 308. The entity table 308 may associate various augmentations from the augmentation table 312 with various images and videos stored in the image table 316 and the video table 314.

[0097] LATENT DIFFUSION MODELS

[0098] In some cases, the example latent diffusion models described herein include a latent feature generator (such as a UNet) and autoencoder, where the autoencoder can be optimized as further described herein. Diffusion models gradually convert a sample x from a real data distribution $p_{\text{data}}(x)$ into a noisy version, (e.g., the diffusion process), and learn to reverse this process by denoising the noisy data step by step. The model transforms a simple distribution (e.g., random Gaussian noise) to a desired more complicated distribution (e.g., real images). Specifically, given a (noise-prediction) diffusion model $\hat{\epsilon}_{\theta}(\cdot)$ parameterized by θ , which is typically structured as a UNet, the training can be formulated as the following noise prediction problem:

$$\min_{\theta} \mathbb{E}_{t \sim U[0,1], x \sim p_{\text{data}}(x), \epsilon \sim \mathcal{N}(0,1)} \|\hat{\epsilon}_{\theta}(t, z_t) - \epsilon\|_2^2,$$

[0099]

[0100] where t refers to the time step; ϵ is the ground-truth noise; $z_t = \alpha_t x + \sigma_t \epsilon$ is the noisy data; α_t and σ_t are the strengths of signal and noise, respectively, decided by a noise scheduler. A trained diffusion model generates samples from noise with various samplers.

For example, DDIM are used to sample with the following iterative denoising process from t to a previous time step t' :

$$z_{t'} = \alpha_{t'} \frac{z_t - \sigma_t \hat{\epsilon}_\theta(t, z_t)}{\alpha_t} + \sigma_{t'} \hat{\epsilon}_\theta(t, z_t),$$

[0101]

[0102] where $z_{t'}$ will be fed into $\hat{\epsilon}_\theta(\cdot)$ again until t' becomes 0 (e.g., the denoising process finishes).

[0103] LATENT DIFFUSION MODELS

[0104] The interaction system disclosed herein applies optimizations to latent diffusion models (LDMs), such as latent diffusion models. Such LDMs reduce the inference computation and steps by performing a denoising process in the latent space, which includes a process of receiving an input that includes noise, and a sequence of steps to transform noisy input progressively to a cleaner version of an image. The latent space is encoded from a pre-trained variational autoencoder (VAE).

[0105] During inference, an image is constructed through the decoder from latent features. The LDM also performs text-to-image generation, where a text prompt embedding c is fed into the diffusion model as a condition.

[0106] When synthesizing images, the interaction system applies a classifier-free guidance to improve quality, such as by applying the following equation:

$$[0107] \quad \tilde{\epsilon}_\theta(t, z_t, c) = w \hat{\epsilon}_\theta(t, z_t, c) - (w - 1) \hat{\epsilon}_\theta(t, z_t, \emptyset),$$

[0108] where $\hat{\epsilon}_\theta(t, z_t, \emptyset)$ represents the unconditional output obtained by using null text $\emptyset$.

The guidance scale w is adjusted to control the strength of conditional information on the generated images to achieve the trade-off between quality and diversity. LDMs are further trained on large-scale datasets, delivering a series of Latent Diffusion models.

[0109] ARCHITECTURE FOR LATENT TEXT-TO-IMAGE DIFFUSION MODEL

[0110] FIG. 4 illustrates an architecture 400 for a latent diffusion model, according to some examples. In the architecture 400, an interaction client 104 inputs a prompt 424 into the latent diffusion model. Latent diffusion models include generative models trained for data generation, restoration, and denoising. These models operate by progressively transforming a noisy or corrupted input into a cleaner version through a sequence of time steps, with each step introducing a controlled level of noise and employing a neural network to denoise the data. This diffusion process allows the model to generate high-quality image data while being conditioned on various levels of noise.

[0111] Features of the architecture of FIG. 4 is described as being performed by or applied to certain systems or applying certain processes, such as a particular machine learning

model (e.g., latent diffusion model), but the processes described herein can be performed by one or more other or the same machine learning models, computer vision models, other models, or a combination thereof.

[0112] In some cases, the interaction system receives a prompt from a user. The

5 interaction client 104 includes a user interface or application that allows a user to input a textual prompt 424. The prompt 424 includes text entered by the user describing desired image characteristics, such as, "a cute puppy sitting in a field". In some cases, the interaction system automatically generates a prompt, such as based on user data as described below.

10 **[0113]** The interaction client 104 identifies keywords from the prompt and applies weights to each of the identified keywords. The interaction client 104 inputs the identified keywords and corresponding weights into a machine learning model.

[0114] In some examples, the interaction client 104 generates a prompt, such as a textual prompt, for the user automatically by identifying an intent based on real-time interaction
15 data captured by the interaction client. For example, the interaction client 104 generates prompts, such as textual prompts, for a user based on a user's past activity, interests, and behavior patterns. The interaction client 104 generates prompts personalized for the user and/or related to topics the user may find appealing, such as if a user frequently interacts with a certain type of content about technology.

20 **[0115]** In some examples, the interaction client 104 uses popular or trending topics from the platform or the wider internet to create prompts that are likely to be of interest to a broad audience. In some examples, by utilizing a user's geographic location, the interaction client 104 can generate prompts that are relevant to the user's local area, such as events, news, or cultural topics. In some examples, the interaction client 104 creates prompts based
25 on the time of day, season, or upcoming events or holidays, such as events that are time sensitive. In some examples, the interaction client 104 uses the user's social connections to generate prompts related to their friends, family, or other users they follow, such as a birthday or new connection with another user. In some examples, based on the user's activity within a specific application or AR experience, the interaction client 104 generates
30 prompts related to that context.

[0116] In some examples, the interaction client 104 uses the user's in-application actions, such as likes, comments, and shares, to generate prompts related to the user's interests. For example, if a user frequently interacts with content about cooking in a recipe application, the interaction client 104 generates a prompt for the user's favorite dish to prepare at
35 home. In some examples, by utilizing sensors and data from the user's mobile device or AR

headset, the interaction client 104 creates context-aware prompts based on their physical environment. In some examples, the interaction client 104 generates prompts based on real-time events occurring within the application or AR experience, such as a live-streamed event. In some examples, the real-time interaction data includes a current camera feed from a camera system of the interaction client 104.

[0117] In some examples, the interaction client 104 uses the user's past activity, preferences, and behavior patterns within the application or AR experience to generate a prompt for the user. In some examples, the interaction client 104 gathers user profile information, such as a calendar of appointments or objects detected in a camera feed of an AR device to generate a prompt. In some examples, by incorporating gamification elements, the interaction client 104 creates prompts that encourage user participation and engagement, such as checking on a feature within a game.

[0118] The prompt 424 is processed by a Vision Transformer (ViT) 426. When the prompt is inputted into the ViT, the text prompt is converted to features in the latent space. For example, the interaction client 104 encodes textual instructions from the textual prompt using the ViT into a representation that can be understood by the latent diffusion model, such as the architecture 400 for the latent diffusion model.

[0119] In some cases, the ViT 426 includes a text encoder. To convert the text prompt to features in the latent space, the text encoder utilizes one or more techniques, such as tokenizing the text prompt into subword units or individual tokens and then mapping them to embedding vectors. These embeddings capture the semantic and contextual information of the text. The text encoder then applies transformer layers to refine the embeddings and generate a representation in the latent space specific to the provided text prompt.

[0120] A latent diffusion model is a probabilistic generative model used in machine learning for tasks like image generation and denoising. The ViT is a specific type of neural network architecture designed for computer vision tasks, and it is particularly well-suited for processing visual data. Although examples described herein explain features as applied to a ViT, it is appreciated that such features can be applied to other transformer-based architectures (e.g., in a stable diffusion model, such as for different types of data or tasks).

[0121] In some cases, a text transformer is used that is pre-trained on textual data for natural language understanding and generation tasks. In some cases, an audio transformer is used that is adapted for audio processing and is trained to work with audio signals and in speech recognition and generation tasks. In some cases, a graph transformer is used when dealing with structured data represented as graphs to model relationships between nodes in the graph.

[0122] In some cases, a time series transformer is used for time series data to handle sequential data effectively and capture temporal dependencies in the data. In some cases, a multi-modal transformer is used that can process and integrate information from different sources. In some cases, custom transformers are used, which are custom designed

5 transformer architectures tailored to specific needs and the model's architecture, attention mechanisms, and input embeddings are adjusted to suit the application.

[0123] By converting text prompts to features in the latent space, the latent diffusion model controls image generation based on textual instructions. The latent diffusion model includes an input layer 438 and an iterative denoising process 416. The input layer 406

10 inputs random noise into the iterative denoising process 416.

[0124] The input layer is a component of the model responsible for receiving random noise. Random noise is essentially a source of randomness that is introduced into the model. This noise is important because the noise allows for diversity in the generated outputs. The noise acts as a bit of randomness to the generation process, ensuring that the latent diffusion

15 model doesn't generate the same output every time the model is used with the same text prompt.

[0125] The iterative denoising process of the latent diffusion model is a multi-step procedure where the model gradually refines the random noise and text-based latent features to generate high-quality images. This process typically involves multiple iterations or steps,

20 where the model incrementally enhances the initial random noise and the latent features from the text prompt. With each step, the generated image becomes clearer and more faithful to the provided textual instructions.

[0126] The latent diffusion model operates in two main stages: forward diffusion and reverse diffusion. These stages are designed to control the generation and quality of images.

25 The forward diffusion stage is the initial step, and it begins with an existing input image.

[0127] In the forward diffusion stage, the input image (such as input image 402) is processed through an encoder (such as encoder 404). The encoder processes the image to extract meaningful features from the input image and represent them in a lower-dimensional latent space. This latent space representation contains essential information about the

30 image's content and characteristics.

[0128] After encoding the input image, the features extracted from the image are then input into an input layer 406. At this stage, the model begins to introduce random noise gradually into the latent features. This introduction of noise is a key aspect of the diffusion process.

[0129] The introduction of noise is progressive and controlled. It starts with a small

35 amount of noise and gradually increases. As the noise level rises, the latent features

representing the image become less informative and more random. This is a deliberate and systematic process designed to transform the encoded image features into a state of complete randomness.

[0130] These features are inputted into an input layer 406 where noise is gradually

introduced until the image becomes complete random noise. The forward diffusion stage continues until the latent features, which initially represented the input image, evolve into a state of complete random noise. In essence, the input image is "diffused" into randomness during this process. The image gradually loses its original content and structure. The noisy image is sent to a buffer 408 and either repeats the addition of noise or sends the finalized noisy image to a buffer 410 for the denoising process.

[0131] In the reverse process, the interaction client 104 gradually removes the predicted noise to recover feature data via an iterative denoising process 416. The iterative denoising process 416 includes an input layer 412, a UNet 414, a Denoising Diffusion Implicit Model (DDIM) scheduler 434, and an output layer 436. The reverse process of a latent diffusion model recovers meaningful feature data from the random noise introduced during the forward diffusion stage. This is achieved through an iterative denoising process.

[0132] The iterative denoising process 416 is repeated by processing data from the input layer 412 through the UNet 414 and to the output layer 436. The DDIM scheduler 434 schedules the iterations of denoising. The denoising process is repeated by updating the time t . This process is repeated several times to gradually remove the predicted noise and recover the original or meaningful features from the data.

[0133] The iterative denoising process begins with an input layer (412). Here, the data from the forward diffusion stage, which includes the random noise and any available information, is provided as input to initiate the denoising process. The UNet includes a neural network architecture commonly used for image segmentation and reconstruction tasks. In the context of the reverse process, the UNet 414 is employed to aid in the reconstruction of meaningful feature data from the noisy input.

[0134] The DDIM scheduler orchestrates the iterations of the denoising process by controlling the timing and frequency of denoising steps. The DDIM scheduler 434 guides the denoising process in a latent diffusion model.

[0135] After processing data through the UNet 414 and other components, the latent diffusion model generates and/or outputs the denoised or recovered feature data at the output layer 436. The output at this stage becomes progressively more refined and closer to the original input as the iterative denoising process continues.

[0136] The denoising process is repeated by updating the parameter representing time, denoted as "t." This updating of time allows the model to control the pace and depth of the denoising at each iteration, ensuring a gradual and controlled recovery of meaningful features.

5 [0137] The reverse process essentially counteracts the forward diffusion by removing the introduced noise and restoring the original or meaningful data. It does so through an iterative and controlled denoising process orchestrated by the DDIM scheduler, with the UNet playing a crucial role in feature recovery. This entire process ensures that the model can generate high-quality and controlled outputs, even when starting from a state of
10 randomness.

[0138] Once a number of iterative denoising processes 416 are completed, the features are outputted from the output layer 436 and into a decoder 418. The decoder is trained to receive features as input and make a prediction 420, such as generating an image 440 that corresponds to the received features.

15 [0139] In latent diffusion models, the UNet 414 is a key component used in the image decoder 418. UNet 414 is a type of convolutional neural network (CNN) architecture that is employed for tasks such as image segmentation and image-to-image translation.

[0140] The UNet architecture includes an encoder-decoder structure with a downsampler 432, a middle layer 430, and an upsampler 428. The downsampler 432 performs
20 downsampling operations, reducing the spatial dimensions of the input image while increasing the number of feature channels. This process extracts hierarchical and abstract features from the input image. Each downsampled layer in the encoder is connected to a corresponding layer in the middle layer 430 and the upsampler 428.

[0141] The upsampler 428 part of the UNet 414 performs upsampling operations to
25 reconstruct features in the latent space by upsampling the encoded features. The decoder 418 of the VAE combines these features with the skip connections to recover spatial details and generate a high-resolution output image. The skip connections aid in preserving fine-grained details by providing additional contextual information to the decoder layers.

[0142] The time-conditional (t) UNet 414 includes machine learning model blocks, such as
30 cross-attention and ResNet blocks. A cross-attention mechanism is employed at each stage to integrate text embedding (c) into spatial features:

[0143]
$$\text{Cross-Attention}(Q_{\mathbf{z}_t}, K_{\mathbf{c}}, V_{\mathbf{c}}) = \text{Softmax}\left(\frac{Q_{\mathbf{z}_t} \cdot K_{\mathbf{c}}^T}{\sqrt{d}}\right) \cdot V_{\mathbf{c}},$$

[0144] where Q is projected from noisy data z_t , K and V are projected from text condition, and d is the feature dimension. UNet also uses ResNet blocks to capture locality. The forwarding of UNet is:

$$\epsilon_\theta(t, z_t) = \prod \{ \text{Cross-Attention}(z_t, c), \text{ResNet}(z_t, t) \}.$$

[0145]

5 [0146] Although examples described herein apply features to UNet, cross-attention blocks, ResNet blocks, and/or a combination thereof, it is appreciated that such features apply to other components of a machine learning model, a latent diffusion model, other types of models processing images or text, and/or the like.

[0147] In some cases, the features described herein apply to U-Net++ which is an
10 extension of the UNet architecture that addresses gradient vanishing and uses nested skip pathways to capture multiscale features more effectively. In some cases, the features apply to SegNet which is an architecture designed for semantic segmentation tasks and uses an encoder-decoder structure with pooling indices for upsampling, which makes it memory-efficient. In some cases, the features apply to PSPNet (Pyramid Scene Parsing Network)
15 which leverages pyramid pooling modules to capture contextual information at multiple scales and is effective for tasks that require understanding the global context of an image.

[0148] In some cases, features are applied to LinkNet which is a lightweight architecture for real-time image segmentation. In some cases, features are applied to a FCN (Fully Convolutional Network) for semantic segmentation by replacing fully connected layers with
20 convolutional layers, enabling end-to-end pixel-wise prediction.

[0149] In some cases, features are applied to self-attention transformers, such as by avoiding cross-attention. In some cases, features are applied to non-local neural networks that introduce non-local operations to capture long-range dependencies in data. In some cases, features are applied to graph Neural Networks (GNNs) that are well-suited for
25 modeling structured data, such as graphs or meshes.

[0150] In some cases, features are applied to spatial transformers that are neural network components that learn to spatially transform feature maps. In some cases, features are applied to convolutional blocks with dilated convolutions which can be used to increase the receptive field of convolutional layers, capturing contextual information without cross-
30 attention.

[0151] In some cases, features are applied to DenseNet which connects each layer to every other layer in a feed-forward fashion helping to alleviate vanishing gradient problems and encouraging feature reuse. In some cases, features are applied to inception modules that use parallel convolutional operations of different kernel sizes to capture multi-scale features

within a single layer. In some cases, features are applied to Xception which is an extension of the Inception architecture that replaces the standard convolutional layers with depthwise separable convolutions.

[0152] Although examples described herein explain a model, such as a stable diffusion model, generating an image, it is appreciated that features described herein also apply to models generating media content items that can include:

- Content augmentations to enhance images, videos, or other media content items to share with others, such as by adjusting the color or appearance or adding interactive elements such as animations and facial transformations, in real-time.
- Emojis that are small images or icons that represent emotions, reactions, or objects.
- Stickers are larger images or animations that can be sent in a chat window.
- Images or photographs can be sent to other users to share visual information or document a particular event.
- Video clips can be used to share recorded content or document a particular event.
- Audio messages can be shared to communicate audible communication.
- Graphics Interchange Formats (GIFs) are short animations that can be used to add humor or express emotions.

[0153] Systems and methods described herein include training a machine learning network, such as training the latent diffusion model described herein. The machine learning network can be trained to generate image from prompts received from the user.

[0154] Training a stable diffusion model includes combining principles of generative modeling and neural network training to enable controlled data generation or manipulation. This training procedure involves two main stages: forward diffusion and reverse diffusion.

[0155] During the forward diffusion stage, an input data point, such as an image, is gradually transformed into random noise by adding noise in a controlled manner. This process begins with a low level of noise and progressively increases it. The noisy data is then encoded using a neural network, capturing its features.

[0156] In the reverse diffusion stage, the encoded noisy data is iteratively denoised to reconstruct the original data. The model learns to remove noise and recover meaningful features guided by a pre-defined loss function, which measures the quality of the reconstruction. This process is repeated iteratively until the data is successfully restored or generated. Training a stable diffusion model requires careful selection of architectural components, loss functions, and optimization techniques to achieve the desired data generation or manipulation capabilities.

[0157] Training of models, such as artificial intelligence models, is necessarily rooted in computer technology, and improves modeling technology by using training data to train such models and thereafter applying the models to new inputs to make inferences on the new inputs.

5 [0158] Such training involves complex processing that typically requires a lot of processor computing and extended periods of time with large training data sets, which are typically performed by massive server systems. Training of models can require logistic regression and/or forward/backward propagating of training data that can include input data and expected output values that are used to adjust parameters of the models. Such training is the
10 framework of machine learning algorithms that enable the models to be applied to new and unseen data (such as new prompt data) and make predictions that the model was trained for based on the weights or scores that were adjusted during training. Such training of the machine learning models described herein reduces false positives and increases the performance of such latent diffusion models.

15 **[0159] CHANGING NODE TOPOLOGY OF DECODER FOR LATENT DIFFUSION**

[0160] FIG. 5 illustrates an example routine 500 for changing the node topology of a decoder for latent diffusion machine learning models, according to some examples. For example, the interaction system changes the node topology of a decoder of the latent diffusion architecture described in FIG. 4. Although the example routine 500 depicts a
20 particular sequence of operations, the sequence may be altered without departing from the scope of the present disclosure. For example, some of the operations depicted may be performed in parallel or in a different sequence that does not materially affect the function of the routine 500. In other examples, different components of an example device or system that implements the routine 500 may perform functions at substantially the same time or in a
25 specific sequence.

[0161] FIG. 5 is described as being performed by certain systems or applying certain processes, such as a latent diffusion machine learning model, but the processes described herein can be performed by one or more other or the same machine learning models.

At block 502, the interaction system identifies an autoencoder for a latent diffusion machine
30 learning model. In some examples, the latent diffusion machine learning model is trained to receive text as input and output an image based on the received text. FIG. 6 illustrates a latent diffusion machine learning model with an autoencoder to convert text to image, according to some examples. The system can generate or receive as input a prompt, such as prompt 620 ("New Super Car"). For example, the decoder of the latent diffusion

architecture of FIG. 4 is used to generate an image from latent features outputted by the UNet.

[0162] Identifying the prompt for the user includes receiving a question or request from the first user via text or speech, as explained above. The interaction system identifies keywords from the prompt and applies weights to each of the identified keywords. The interaction system applies the identified keywords and corresponding weights to a machine learning model.

[0163] In some examples, the interaction system generates the prompt for the user automatically based on an intent identified from real-time interaction data captured by the interaction client. The interaction system generates prompts for a user based on a user's past activity, interests, and behavior patterns. The interaction system generates personalized prompts related to topics the user may find appealing, such as if a user frequently interacts with a certain type of content about technology.

[0164] In some examples, the interaction system uses popular or trending topics from the platform or the wider internet to create prompts that are likely to be of interest to a broad audience. In some examples, by utilizing a user's geographic location, the interaction system can generate prompts that are relevant to their local area, such as events, news, or cultural topics. In some examples, the interaction system can create prompts based on the time of day, season, or upcoming events or holidays, such as events that are time sensitive. In some examples, the interaction system can use the user's social connections to generate prompts related to their friends, family, or other users they follow, such as a birthday or new connection with another user. In some examples, based on the user's activity within a specific application or AR experience, the interaction system can generate prompts related to that context.

[0165] In some examples, the interaction system can use the user's in-application actions, such as likes, comments, and shares, to generate prompts related to their interests. For example, if a user frequently interacts with content about cooking in a recipe application, the interaction system may generate a prompt for the user's favorite dish to prepare at home. In some examples, by utilizing sensors and data from the user's mobile device or AR headset, the interaction system creates context-aware prompts based on their physical environment. In some examples, the interaction system can generate prompts based on real-time events occurring within the application or AR experience, such as a live-streamed event. In some examples, the real-time interaction data includes a current camera feed from a camera system of the first interaction client 104.

[0166] In some examples, the interaction system uses the user's past activity, preferences, and behavior patterns within the application or AR experience to generate a prompt for the user. In some examples, the interaction system gathers user profile information, such as a calendar of appointments or objects detected in a camera feed of an AR device to generate a prompt. In some examples, by incorporating gamification elements, the interaction system creates prompts that encourage user participation and engagement, such as checking on a feature within a game.

[0167] The interaction system inputs the received or generated prompt to the latent diffusion model 602. The latent diffusion model (such as the latent diffusion model described in FIG. 4), generates latent features. Such latent features are inputted into the decoder. The decoder outputs an image 618 based on the received latent features.

[0168] Returning to FIG. 5, at block 504, the interaction system identifies a number of channels in a decoder of the autoencoder. In some examples, the decoder is configured to receive latent features as input and output one or more images. In the example in FIG. 6, the interaction system identifies 5 channels in the decoder, a first channel 604, a second channel 606, a third channel 608, a fourth channel 610, and a fifth channel 612.

[0169] At block 506, the interaction system identifies a performance characteristic of the decoder. In some cases, a performance characteristic of the decoder is identified to determine how to change the node or channel topology of the decoder. For example, a performance characteristic such as the magnitude of each wave corresponding to a particular layer can be used to identify whether to keep or remove a channel. Each channel can output certain waves.

[0170] In some cases, the output of one or more channels are normalized (e.g., a mean or standard deviation), and the learned parameters for the normalized layer corresponds to the magnitude of the output for that channel. In some cases, if the magnitude is larger than a certain maximum threshold then the channel is kept, if the magnitude is lower than a certain minimal threshold then the channel is removed, and/or vice versa. In some cases, the channels represent paths for data to be processed through nodes. In some cases, the channels represent convolutional filters.

[0171] In some cases, a performance characteristic includes a reconstruction loss that measures how well the autoencoder can reconstruct the input data, indicating the quality of the learned representations. The interaction system can identify a latent space dimensionality that determines the optimal dimensionality of the latent space helps balance representation capacity and compression.

[0172] The interaction system can identify performance characteristics as described below, such as a denoising capability assessing how effectively the autoencoder can remove noise from input data, which is crucial for various applications. The interaction system can identify a transferability metric evaluating the extent to which the learned representations can be transferred to other tasks or domains.

[0173] The interaction system can identify an anomaly detection that detects anomalies by identifying data points that lead to high reconstruction error. The interaction system can identify a data generation metric gauging the model's generative capabilities of generating new data samples by sampling from the latent space.

[0174] The interaction system can identify an interpolation metric evaluating the smoothness of interpolation in the latent space, assessing the model's ability to generate meaningful transitions between data points. The interaction system can identify a feature learning metric that determines the autoencoder's capacity to learn meaningful features or patterns from the input data.

[0175] The interaction system can identify a robustness to adversarial attacks measuring the resistance of the autoencoder to adversarial perturbations in the input data. The interaction system can identify a convergence speed assessing how quickly the autoencoder converges during training, which can impact practical usability.

[0176] The interaction system can identify a scalability metric examining how well the model performs as the dataset size and complexity increase. The interaction system can identify a regularization effectiveness analyzing the ability of the autoencoder to implicitly apply regularization to prevent overfitting. The interaction system can identify an energy efficiency evaluating the model's computational efficiency and resource requirements.

[0177] The interaction system can identify a model size assessing the trade-off between model size and performance in terms of memory and computation. The interaction system can identify an inference speed measuring the time it takes for the autoencoder to process a single input.

[0178] The interaction system can identify a generalization metric evaluating how well the autoencoder generalizes to unseen data and whether it can capture the underlying data distribution. The interaction system can identify a quantization effects measuring how quantization of the latent space affects model performance and quality of reconstructions.

[0179] The interaction system can identify a privacy-preserving properties assessing the extent to which the model preserves the privacy of the input data. The interaction system can identify an explainability evaluating the interpretability of the learned representations in the latent space. The interaction system can identify a sensitivity to hyperparameters

evaluating how changes in hyperparameters, such as learning rates and batch sizes, impact the model's performance.

[0180] In some cases, the features described herein, such as identifying the performance characteristic or changing the topology is applied to other parts of the architecture, such as the UNet of the latent diffusion model or the encoder. For example, the performance characteristic of the encoder can be used to add or remove blocks, such as a cross attention block or ResNet block of the latent diffusion model.

[0181] Although the examples described herein refer to removing a channel, it is appreciated that such features can include adding a channel, adding or removing nodes, adding or removing blocks, and/or the like.

[0182] At block 508, the interaction system changes the node or channel topology of the decoder based on the performance characteristic to generate an updated decoder. In some cases, the interaction system changes the topology by removing a node, such as eliminating a single unit or neuron from a layer in the decoder, reducing the model's capacity and potentially simplifying the learned representations.

[0183] In some cases, the interaction system removes an entire layer from the decoder architecture which can significantly reduce the model's depth, potentially making it less expressive and computationally lighter.

[0184] In some cases, the interaction system removes a channel. In convolutional decoder architectures, removing a channel from a layer includes reducing the number of filters, which can affect the complexity of the learned features and representations. In FIG. 6, channels 606 and 610 are highlighted as outputting a magnitude below a certain threshold and are identified to be removed.

[0185] In some cases, the interaction system adjusts layer width, such as by changing the number of nodes (neurons) in a layer, (e.g., increasing or decreasing the width), that can impact the model's capacity and influence its ability to capture complex patterns in the data. In some cases, the interaction system skips connections, such as by adding or removing skip connections between layers in the decoder that can affect the flow of information and gradients, potentially improving or simplifying the model's training and performance.

[0186] In some cases, the interaction system changes activation functions, such as by altering the activation functions in the decoder that can impact the non-linearity of the model and its ability to approximate complex functions. In some cases, the interaction system performs weight sharing, such as by introducing weight sharing between layers that can reduce the number of learnable parameters and potentially lead to more robust representations.

[0187] In some cases, the interaction system adds or removes recurrent connections, such as for sequential data, by adding or removing recurrent connections in the decoder that can influence its ability to capture temporal dependencies. In some cases, the interaction system changes connectivity patterns, such as by modifying the connections between nodes (e.g., introducing or removing connections in a graph-based decoder) that can alter the information flow and the model's capacity.

[0188] In some cases, the interaction system performs normalization techniques, such as by applying or removing normalization techniques (e.g., batch normalization or layer normalization), that can impact the stability and training dynamics of the decoder.

[0189] In some cases, the interaction system can vary kernel sizes, such as in convolutional decoders, by changing the kernel sizes in different layers that can affect the scale of features that the model can capture. In some cases, the interaction system adds attention mechanisms, such as by incorporating attention mechanisms in the decoder that can help it focus on relevant parts of the input and potentially improve performance.

[0190] FIG. 7 illustrates an updated decoder with removed channels and newly added channels, according to some examples. In FIG. 7, channels 606 and 610 from FIG. 6 are removed and new channels 702 and 704 are added. In some examples, the interaction system identifies the channels that perform best based on certain performance characteristic and copy those channels to add back in. In other examples, a new channel includes a channel with uniform weights and characteristics for an untrained channel.

[0191] After removing channels or nodes from a decoder in a latent diffusion model for optimization, the interaction system assesses one or more characteristics to determine whether it is beneficial to add channels or nodes back into the decoder.

[0192] In some cases, the interaction system determines reconstruction quality to evaluate how well the reduced decoder can reconstruct the input data. If the reconstruction quality is noticeably degraded, the interaction system can add nodes or channels to increase capacity.

[0193] In some cases, the interaction system checks for overfitting, such as checking for signs of overfitting on the training data. If overfitting is observed, the interaction system increases model capacity to capture more complex patterns in the data.

[0194] In some cases, the interaction system checks for underfitting. If the model is underfitting the data (e.g., the model cannot capture the essential features), the interaction system adds channels or nodes back to help the model better fit the training data.

[0195] In some cases, the interaction system assesses a loss function. If the loss function converges to a suboptimal or higher value after capacity reduction, the interaction system adds nodes or channels for more capacity for the decoder.

[0196] In some cases, the interaction system assesses generalization performance, such as by assessing the model's performance on a validation or test dataset. If the performance deteriorates significantly, the interaction system adds nodes or channels to increase the model's capacity to generalize effectively.

5 **[0197]** In some cases, the interaction system reduces the channel dimensions of at least one layer of the image decoder to obtain a compressed image decoder. The compressed image decoder can be of lower latency and have fewer parameters compared to the image decoder. The interaction system identifies a balance between performance metrics, such as between (1) latency and resource efficiency and (2) image quality and representational capacity.

10 **[0198]** In some cases, the adding and/or removing in the node topology changes are repeated until a certain optimization characteristic (characteristics described herein) is achieved. For example, channels are removed until a certain latency metric is met and channels are added until a certain performance threshold is met. The channels are removed and added until both the latency metric and performance threshold are both met.

15 **[0199]** At block 510, the interaction system retrains the latent diffusion machine learning model using the updated decoder. For example, the interaction system retrains the latent diffusion machine learning model by first inputting latent features to the updated decoder.

[0200] FIG. 8 illustrates a comparison of the prior version of the decoder and the updated decoder for retraining, according to some examples. For example, the decoder of FIG. 6 and the decoder of FIG. 7 are compared. The original decoder still has channels 606 and 610, whereas the updated decoder has channels 606 and 610 removed and new channels 702 and 704 added.

25 **[0201]** During retraining of the updated decoder, synthetic data is generated by inputting noise 806 to the latent diffusion model 804. The noise inputted into the latent diffusion model generates latent features outputted by the UNet of the latent diffusion model. The latent features are then inputted to the updated decoder to generate an output image.

[0202] The interaction system then assesses an outputted image from the updated decoder. Based on the outputted image, the interaction system updates one or more weights of the decoder based on the assessment of the outputted image.

30 **[0203]** In some cases, the interaction system inputs the latent features into both the original decoder and the updated decoder. The output of the original decoder and the updated decoder are assessed by the comparison and selection module 802.

[0204] In some cases, the comparison and selection module 802 compares the outputs of the original decoder and the updated decoder to make changes to the decoder. For example, a mean squared error can be determined between the two to determine how well the updated

35

decoder is performing as compared with the original decoder. The interaction system updates the weights of the updated decoder and reruns the synthetic training data in order to minimize the mean squared error.

[0205] In some cases, the interaction system compares the outputted images using a mean Squared Error (MSE) that calculates the average of the squared differences between corresponding pixels in two images. The MSE provides a measure of the overall pixel-wise difference between the images, with higher values indicating greater dissimilarity.

[0206] In some cases, the interaction system compares the outputted images via a Structural Similarity Index (SSIM) that assesses the structural similarity by considering luminance, contrast, and structure. SSIM provides a more comprehensive measure of image dissimilarity, accounting for human perceptual differences.

[0207] In some cases, the interaction system compares the outputted images using a Peak Signal-to-Noise Ratio (PSNR) that quantifies the quality of image reconstruction by measuring the ratio of the maximum possible power of a signal (the original image) to the power of corrupting noise (the difference between the images). Higher PSNR values indicate better image quality.

[0208] In some cases, the interaction system compares the outputted images using cross-correlation that measures the similarity between two images by sliding one image over the other and computing the dot product at each position. High cross-correlation indicates similarity, while low values suggest dissimilarity.

[0209] In some cases, the interaction system compares the outputted images using a histogram comparison that compares histograms of pixel values in two images can reveal differences in color or intensity distributions. In some cases, the interaction system compares the outputted images using perceptual metrics that take into account human perception and assess how different the images appear to the human eye.

[0210] In some cases, the interaction system compares the outputted images using feature-based comparisons by extracting features from images using techniques like deep neural networks and comparing those feature representations can capture more abstract differences between images.

[0211] Although examples described herein explain the generation of an image, but it is appreciated that the image can be a frame of a video and/or the features described herein can be applied to a video, a three dimensional space, virtual images, video, or 3D objects, media content items, and/or the like.

[0212] The media content items include:

- Content augmentations to enhance images, videos, or other media content items to share with others, such as by adjusting the color or appearance or adding interactive elements such as animations and facial transformations, in real-time.
- Emojis that are small images or icons that represent emotions, reactions, or objects.
- 5 • Stickers are larger images or animations that can be sent in a chat window.
- Images or photographs can be sent to other users to share visual information or document a particular event.
- Video clips can be used to share recorded content or document a particular event.
- Audio messages can be shared to communicate audible communication.
- 10 • Graphics Interchange Formats (GIFs) are short animations that can be used to add humor or express emotions.

[0213] Systems and methods described herein include training a machine learning network, such as training to generate images. The machine learning network can be trained to receive text prompts and generate different images through each iteration of inferences. The machine learning algorithm can be trained using synthetic data from random noise.

[0214] Training of models, such as artificial intelligence models, is necessarily rooted in computer technology, and improves modeling technology by using training data to train such models and thereafter applying the models to new inputs to make inferences on the new inputs. Here, the new inputs can be new prompts generated based on user data or received from a user. The trained machine learning model can generate a desired image based on the prompt data.

[0215] Such training involves complex processing that typically requires a lot of processor computing and extended periods of time with large training data sets, which are typically performed by massive server systems. Training of models can require logistic regression and/or forward/backward propagating of training data that can include input data and expected output values that are used to adjust parameters of the models. Such training is the framework of machine learning algorithms that enable the models to be applied to new and unseen data (such as new interaction data) and make predictions that the model was trained for based on the weights or scores that were adjusted during training. Such training of the machine learning models described herein reduces false positives and increases the performance.

[0216] FIG. 9 illustrates removing and adding channels efficiently to quickly identify an optimal decoder architecture, according to some examples. For example, the interaction system removes and adds channels in the decoder of the latent diffusion architecture of FIG. 4. The original decoder at stage 920 can include 20 channels. Half of the channels can be

selected to be removed. In some cases, performance characteristics for the channels are identified and ranked from top to bottom. In some cases, the channels are ranked without changing the architecture of the decoder.

[0217] The middle ranked channel 908 is identified at stage 922, and the channels with the lowest performance characteristic values (such as magnitude of the output) are removed, such as channels 1-9 at stage 924. Then another performance characteristic is assessed (such as latency for the entire decoder). If the compressed decoder at stage 924 with channels 10-20 still does not meet the desired latency value, the middle channel again is identified, such as channel 912.

[0218] Again the lower performing channels are removed (e.g., channels 10-14). At stage 926, the desired latency is achieved. The system then assesses the decoder performance. The decoder at stage 926 does not meet the desired performance requirements. Then the system identifies the channel that is best performing, such as channel 930. At stage 928, this channel is added in various parts of the decoder, such as where the magnitude of the output is lowest to improve the performance requirement. The process is repeated until the latency and performance requirements are both met to generate an optimal decoder.

[0219] DECODER ARCHITECTURE OPTIMIZATION

[0220] In an autoencoder latent diffusion machine learning model, the decoder is responsible for reconstructing the input data from the encoded representation. The decoder typically includes of multiple layers that gradually upsample and transform the encoded features to produce the final output. Channels in the decoder refer to the number of feature maps or image channels present in each layer of the decoder network.

[0221] In traditional latent diffusion models, the image decoder typically includes of a large number of channels. The interaction clients 104 described herein reduce the number of channels, resulting in a more efficient decoder (referred to herein as modified decoder or retrained decoder). In some cases, the interaction client 104 focuses only on optimizing the decoder and does not optimize the image encoder, which is part of the VAE (Variational Autoencoder) framework, and/or vice versa.

[0222] In some cases, another characteristic of the decoder is modified, such as a number of layers. FIG. 10 illustrates a modified decoder 1000, according to some examples. For example, the interaction system modifies the decoder of the latent diffusion architecture of FIG. 4. The decoder 1002 includes a number of channels and layers for the decoder. The decoder 1002 is modified to generate a modified decoder 1004 with a smaller number of channels.

[0223] To retrain the decoder, the interaction client 104 identifies features in the latent space for input text prompts. The interaction client 104 inputs this latent representation of features into both the modified decoder with less channels and the original decoder to generate two images.

5 **[0224]** The interaction client 104 compares the image generated by the original decoder and the modified decoder. For example, the interaction client 104 identifies a mean squared error between the two images. By comparing the outputs of the modified decoder and the original decoder, the interaction client 104 adjusts parameters of the modified decoder to minimize the discrepancies between the images.

10 **[0225]** The interaction client 104 applies synthetic data to generate the modified image decoder. Using synthetic data for distillation enables the interaction client 104 to augment the training dataset on-the-fly where each prompt is used to obtain any amount of images by sampling various noises.

[0226] In the context of latent diffusion models, synthetic data refers to artificially
15 generated data that is used to enhance the training process. The interaction client 104 creates synthetic data using computational techniques, such as algorithms or simulations, rather than being directly collected from real-world sources. This allows for the generation of data that may not exist or is difficult to obtain in reality.

[0227] The interaction client 104 creates synthetic data to expand the training dataset by
20 creating additional samples that exhibit different variations, complexities, or scenarios. By introducing a diverse range of synthetic data, the model becomes more robust and capable of handling various input conditions and generating more accurate and reliable outputs. For example, the interaction client 104 generates noise data, such as random noise data, and applies the same or similar prompts in order to generate different images using latent
25 diffusion.

[0228] The interaction client 104 applies synthetic data for distillation, where the data is employed to train and optimize the modified image decoder. By generating synthetic data on-the-fly during distillation, the interaction client creates an unlimited number of training examples for each text prompt by sampling various noises. This data enhances the training
30 process, enriches the dataset, and allows for the optimization of the image decoder based on a wider range of scenarios and variations.

[0229] In some cases, the interaction client 104 identifies a prompt of a user, the prompt indicative of an intent of the user for generative images. The interaction client 104 accesses noise data and analyzes a collection of data corresponding to the prompt and the noise data
35 using a machine learning model to generate features. The interaction client 104 then

analyzes the features using a first decoder to generate a first image corresponding to the features.

[0230] In some cases, the interaction client 104 removes a number of channels from the first decoder to generate a second decoder, analyzes the features using the second decoder to generate a second image corresponding to the features, and comparing the first image and second image. The interaction client 104 then modifies the second decoder based on the comparison.

[0231] In some cases, the interaction client 104 reduces a number of channels of the decoder. In some cases, the interaction client 104 reduces a number of layers of the decoder. The interaction client 104 reduces number by half, a quarter, or other fraction of the original number. In some cases, the interaction client 104 reduces the number in increments, such as by 1 layer or 2 layers.

[0232] Examples described herein refer to a reduction of a particular characteristic of the decoder, such as a channel or layer. However, it is appreciated that other characteristics apply to the features and examples described herein.

[0233] In some cases, the interaction client 104 compares the output of the original decoder and the modified decoder using a comparison function. The comparison function is a mean squared error, an average, a peak, and/or the like.

[0234] In some cases, the interaction client 104 determines based on the comparison whether to continue with the currently modified decoder. In the subsequent optimization, the interaction client 104 reduces further a number of layers and/or channels from the modified decoder. The interaction client 104 compares the output of the further modified decoder with the parent modified decoder, the original decoder, and/or a decoder version between.

[0235] In some cases, the interaction client reduces the number of channels by a certain fraction, such as half, compares the outputs, and decides whether to add or reduce the number of channels thereof. For example, a 32 channel decoder can be compared with a 16 channel encoder. If the performance of the 16 channel decoder meets a certain threshold, the interaction client 104 continues removing channels to an 8 channel decoder. On the other hand, if the performance of the channel decoder does not meet the threshold, the interaction client 104 adds channels to a 24 channel decoder. Advantageously, the optical decoder is found using $\log(n)$ number or modifications.

[0236] In some cases, the interaction client 104 applies synthetic data when training the decoder as described herein. The input to the decoders can be the same or different. The

synthetic data includes random noise, such as a Gaussian noise distribution. As such, an unlimited number of training data can be applied to this system.

[0237] In some cases, the process above including modifying of decoders and comparing outputs is repeated until a certain performance threshold is met. The performance threshold includes latency, memory, size, and/or the like. In some cases, the performance threshold includes a balance between individual performance metrics.

[0238] In some cases, the interaction client 104 applies modifications to the decoder while still maintaining certain characteristics. In some cases, the interaction client 104 modifies the number of layers or channels while still maintaining the output image resolution.

[0239] In some cases, the original decoder has already been trained and the processes described herein retrains the already trained original decoder. The interaction client 104 retrains the decoder via the process of modifying and comparing using synthetic data as described herein.

[0240] In some cases, the interaction client 104 retrains an encoder with or without the decoder. In some cases, the modifications to the encoder is the same as the decoder. The interaction client 104 make modifications individually or at the same time with the encoder and decoder through each step of the retraining process.

[0241] Examples described herein refer to the encoder and/or decoder, but it is appreciated that the features and examples apply to either the encoder and/or the decoder.

[0242] DATA COMMUNICATIONS ARCHITECTURE

[0243] FIG. 11 is a schematic diagram illustrating a structure of a message 1100, according to some examples, generated by an interaction client 104 for communication to a further interaction client 104 via the interaction servers 124. The content of a particular message 1100 is used to populate the message table 306 stored within the database 304, accessible by the interaction servers 124. Similarly, the content of a message 1100 is stored in memory as "in-transit" or "in-flight" data of the user system 102 or the interaction servers 124. A message 1100 is shown to include the following example components:

- Message identifier 1102: a unique identifier that identifies the message 1100.
- Message text payload 1104: text, to be generated by a user via a user interface of the user system 102, and that is included in the message 1100.
- Message image payload 1106: image data, captured by a camera component of a user system 102 or retrieved from a memory component of a user system 102, and that is included in the message 1100. Image data for a sent or received message 1100 may be stored in the image table 316.

- Message video payload 1108: video data, captured by a camera component or retrieved from a memory component of the user system 102, and that is included in the message 1100. Video data for a sent or received message 1100 may be stored in the image table 316.

5 • Message audio payload 1110: audio data, captured by a microphone or retrieved from a memory component of the user system 102, and that is included in the message 1100.

10 • Message augmentation data 1112: augmentation data (e.g., filters, stickers, or other annotations or enhancements) that represents augmentations to be applied to message image payload 1106, message video payload 1108, or message audio payload 1110 of the message 1100. Augmentation data for a sent or received message 1100 may be stored in the augmentation table 312.

15 • Message duration parameter 1114: parameter value indicating, in seconds, the amount of time for which content of the message (e.g., the message image payload 1106, message video payload 1108, message audio payload 1110) is to be presented or made accessible to a user via the interaction client 104.

20 • Message geolocation parameter 1116: geolocation data (e.g., latitudinal and longitudinal coordinates) associated with the content payload of the message. Multiple message geolocation parameter 1116 values may be included in the payload, each of these parameter values being associated with respect to content items included in the content (e.g., a specific image within the message image payload 1106, or a specific video in the message video payload 1108).

25 • Message story identifier 1118: identifier values identifying one or more content collections (e.g., "stories" identified in the collections table 318) with which a particular content item in the message image payload 1106 of the message 1100 is associated. For example, multiple images within the message image payload 1106 may each be associated with multiple content collections using identifier values.

30 • Message tag 1120: each message 1100 may be tagged with multiple tags, each of which is indicative of the subject matter of content included in the message payload. For example, where a particular image included in the message image payload 1106 depicts an animal (e.g., a lion), a tag value may be included within the message tag 1120 that is indicative of the relevant animal. Tag values may be generated manually, based on user input, or may be automatically generated using, for example, image recognition.

35 • Message sender identifier 1122: an identifier (e.g., a messaging system identifier, email address, or device identifier) indicative of a user of the user system 102 on which the message 1100 was generated and from which the message 1100 was sent.

- Message receiver identifier 1124: an identifier (e.g., a messaging system identifier, email address, or device identifier) indicative of a user of the user system 102 to which the message 1100 is addressed.

[0244] The contents (e.g., values) of the various components of message 1100 may be pointers to locations in tables within which content data values are stored. For example, an image value in the message image payload 1106 may be a pointer to (or address of) a location within an image table 316. Similarly, values within the message video payload 1108 may point to data stored within an image or video table 316, values stored within the message augmentation data 1112 may point to data stored in an augmentation table 312, values stored within the message story identifier 1118 may point to data stored in a collections table 318, and values stored within the message sender identifier 1122 and the message receiver identifier 1124 may point to user records stored within an entity table 308.

[0245] SYSTEM WITH HEAD-WEARABLE APPARATUS

[0246] FIG. 12 illustrates a system 1200 including a head-wearable apparatus 116 with a selector input device, according to some examples. FIG. 12 is a high-level functional block diagram of an example head-wearable apparatus 116 communicatively coupled to a mobile device 114 and various server systems 1204 (e.g., the interaction server system 110) via various networks 108. The networks 108 may include any combination of wired and wireless connections.

[0247] The head-wearable apparatus 116 includes one or more cameras, each of which may be, for example, a visible light camera 1206, an infrared emitter 1208, and an infrared camera 1210.

[0248] An interaction client, such as a mobile device 114 connects with head-wearable apparatus 116 using both a low-power wireless connection 1212 and a high-speed wireless connection 1214. The mobile device 114 is also connected to the server system 1204 and the network 1216.

[0249] The head-wearable apparatus 116 further includes two image displays of the image display of optical assembly 1218. The two image displays of optical assembly 1218 include one associated with the left lateral side and one associated with the right lateral side of the head-wearable apparatus 116. The head-wearable apparatus 116 also includes an image display driver 1220, an image processor 1222, low-power circuitry 1224, and high-speed circuitry 1226. The image display of optical assembly 1218 is for presenting images and videos, including an image that can include a graphical user interface to a user of the head-wearable apparatus 116.

[0250] The image display driver 1220 commands and controls the image display of optical assembly 1218. The image display driver 1220 may deliver image data directly to the image display of optical assembly 1218 for presentation or may convert the image data into a signal or data format suitable for delivery to the image display device. For example, the image data may be video data formatted according to compression formats, such as H.264 (MPEG-4 Part 10), HEVC, Theora, Dirac, RealVideo RV40, VP8, VP9, or the like, and still image data may be formatted according to compression formats such as Portable Network Group (PNG), Joint Photographic Experts Group (JPEG), Tagged Image File Format (TIFF) or exchangeable image file format (EXIF) or the like.

[0251] The head-wearable apparatus 116 includes a frame and stems (or temples) extending from a lateral side of the frame. The head-wearable apparatus 116 further includes a user input device 1228 (e.g., touch sensor or push button), including an input surface on the head-wearable apparatus 116. The user input device 1228 (e.g., touch sensor or push button) is to receive from the user an input selection to manipulate the graphical user interface of the presented image.

[0252] The components shown in FIG. 12 for the head-wearable apparatus 116 are located on one or more circuit boards, for example a PCB or flexible PCB, in the rims or temples. Alternatively, or additionally, the depicted components can be located in the chunks, frames, hinges, or bridge of the head-wearable apparatus 116. Left and right visible light cameras 1206 can include digital camera elements such as a complementary metal oxide–semiconductor (CMOS) image sensor, charge-coupled device, camera lenses, or any other respective visible or light-capturing elements that may be used to capture data, including images of scenes with unknown objects.

[0253] The head-wearable apparatus 116 includes a memory 1202, which stores instructions to perform a subset or all of the functions described herein. The memory 1202 can also include storage device.

[0254] As shown in FIG. 12, the high-speed circuitry 1226 includes a high-speed processor 1230, a memory 1202, and high-speed wireless circuitry 1232. In some examples, the image display driver 1220 is coupled to the high-speed circuitry 1226 and operated by the high-speed processor 1230 in order to drive the left and right image displays of the image display of optical assembly 1218. The high-speed processor 1230 may be any processor capable of managing high-speed communications and operation of any general computing system needed for the head-wearable apparatus 116. The high-speed processor 1230 includes processing resources needed for managing high-speed data transfers on a high-speed wireless connection 1214 to a wireless local area network (WLAN) using the

high-speed wireless circuitry 1232. In certain examples, the high-speed processor 1230 executes an operating system such as a LINUX operating system or other such operating system of the head-wearable apparatus 116, and the operating system is stored in the memory 1202 for execution. In addition to any other responsibilities, the high-speed processor 1230 executing a software architecture for the head-wearable apparatus 116 is used to manage data transfers with high-speed wireless circuitry 1232. In certain examples, the high-speed wireless circuitry 1232 is configured to implement Institute of Electrical and Electronic Engineers (IEEE) 802.11 communication standards, also referred to herein as WI-FI®. In some examples, other high-speed communications standards may be implemented by the high-speed wireless circuitry 1232.

[0255] The low-power wireless circuitry 1234 and the high-speed wireless circuitry 1232 of the head-wearable apparatus 116 can include short-range transceivers (Bluetooth™) and wireless wide, local, or wide area network transceivers (e.g., cellular or WI-FI®). Mobile device 114, including the transceivers communicating via the low-power wireless connection 1212 and the high-speed wireless connection 1214, may be implemented using details of the architecture of the head-wearable apparatus 116, as can other elements of the network 1216.

[0256] The memory 1202 includes any storage device capable of storing various data and applications, including, among other things, camera data generated by the left and right visible light cameras 1206, the infrared camera 1210, and the image processor 1222, as well as images generated for display by the image display driver 1220 on the image displays of the image display of optical assembly 1218. While the memory 1202 is shown as integrated with high-speed circuitry 1226, in some examples, the memory 1202 may be an independent standalone element of the head-wearable apparatus 116. In certain such examples, electrical routing lines may provide a connection through a chip that includes the high-speed processor 1230 from the image processor 1222 or the low-power processor 1236 to the memory 1202. In some examples, the high-speed processor 1230 may manage addressing of the memory 1202 such that the low-power processor 1236 will boot the high-speed processor 1230 any time that a read or write operation involving memory 1202 is needed.

[0257] As shown in FIG. 12, the low-power processor 1236 or high-speed processor 1230 of the head-wearable apparatus 116 can be coupled to the camera (visible light camera 1206, infrared emitter 1208, or infrared camera 1210), the image display driver 1220, the user input device 1228 (e.g., touch sensor or push button), and the memory 1202.

[0258] The head-wearable apparatus 116 is connected to a host computer. For example, the head-wearable apparatus 116 is paired with the mobile device 114 via the high-speed

wireless connection 1214 or connected to the server system 1204 via the network 1216. The server system 1204 may be one or more computing devices as part of a service or network computing system, for example, that includes a processor, a memory, and network communication interface to communicate over the network 1216 with the mobile device 114 and the head-wearable apparatus 116.

[0259] The mobile device 114 includes a processor and a network communication interface coupled to the processor. The network communication interface allows for communication over the network 1216, low-power wireless connection 1212, or high-speed wireless connection 1214. Mobile device 114 can further store at least portions of the instructions in the mobile device 114's memory to implement the functionality described herein.

[0260] Output components of the head-wearable apparatus 116 include visual components, such as a display such as a liquid crystal display (LCD), a plasma display panel (PDP), a light-emitting diode (LED) display, a projector, or a waveguide. The image displays of the optical assembly are driven by the image display driver 1220. The output components of the head-wearable apparatus 116 further include acoustic components (e.g., speakers), haptic components (e.g., a vibratory motor), other signal generators, and so forth. The input components of the head-wearable apparatus 116, the mobile device 114, and server system 1204, such as the user input device 1228, may include alphanumeric input components (e.g., a keyboard, a touch screen configured to receive alphanumeric input, a photo-optical keyboard, or other alphanumeric input components), point-based input components (e.g., a mouse, a touchpad, a trackball, a joystick, a motion sensor, or other pointing instruments), tactile input components (e.g., a physical button, a touch screen that provides location and force of touches or touch gestures, or other tactile input components), audio input components (e.g., a microphone), and the like.

[0261] The head-wearable apparatus 116 may also include additional peripheral device elements. Such peripheral device elements may include biometric sensors, additional sensors, or display elements integrated with the head-wearable apparatus 116. For example, peripheral device elements may include any I/O components including output components, motion components, position components, or any other such elements described herein.

[0262] For example, the biometric components include components to detect expressions (e.g., hand expressions, facial expressions, vocal expressions, body gestures, or eye-tracking), measure biosignals (e.g., blood pressure, heart rate, body temperature, perspiration, or brain waves), identify a person (e.g., voice identification, retinal

identification, facial identification, fingerprint identification, or electroencephalogram based identification), and the like.

[0263] The motion components include acceleration sensor components (e.g., accelerometer), gravitation sensor components, rotation sensor components (e.g.,

gyroscope), and so forth. The position components include location sensor components to generate location coordinates (e.g., a Global Positioning System (GPS) receiver component), Wi-Fi or Bluetooth™ transceivers to generate positioning system coordinates, altitude sensor components (e.g., altimeters or barometers that detect air pressure from which altitude may be derived), orientation sensor components (e.g., magnetometers), and the like. Such positioning system coordinates can also be received over low-power wireless connections 1212 and high-speed wireless connection 1214 from the mobile device 114 via the low-power wireless circuitry 1234 or high-speed wireless circuitry 1232.

[0264] MACHINE ARCHITECTURE

[0265] FIG. 13 is a diagrammatic representation of the machine 1300 within which

instructions 1302 (e.g., software, a program, an application, an applet, an app, or other executable code) for causing the machine 1300 to perform any one or more of the methodologies discussed herein may be executed. For example, the instructions 1302 may cause the machine 1300 to execute any one or more of the methods described herein. The instructions 1302 transform the general, non-programmed machine 1300 into a particular machine 1300 programmed to carry out the described and illustrated functions in the manner described. The machine 1300 may operate as a standalone device or may be coupled (e.g., networked) to other machines. In a networked deployment, the machine 1300 may operate in the capacity of a server machine or a client machine in a server-client network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine 1300 may comprise, but not be limited to, a server computer, a client computer, a personal computer (PC), a tablet computer, a laptop computer, a netbook, a set-top box (STB), a personal digital assistant (PDA), an entertainment media system, a cellular telephone, a smartphone, a mobile device, a wearable device (e.g., a smartwatch), a smart home device (e.g., a smart appliance), other smart devices, a web appliance, a network router, a network switch, a network bridge, or any machine capable of executing the instructions 1302, sequentially or otherwise, that specify actions to be taken by the machine 1300. Further, while a single machine 1300 is illustrated, the term "machine" shall also be taken to include a collection of machines that individually or jointly execute the instructions 1302 to perform any one or more of the methodologies discussed herein. The machine 1300, for example, may comprise the user system 102 or any one of multiple server

devices forming part of the interaction server system 110. In some examples, the machine 1300 may also comprise both client and server systems, with certain operations of a particular method or algorithm being performed on the server-side and with certain operations of the particular method or algorithm being performed on the client-side.

5 **[0266]** The machine 1300 may include processors 1304, memory 1306, and input/output I/O components 1308, which may be configured to communicate with each other via a bus 1310. In an example, the processors 1304 (e.g., a Central Processing Unit (CPU), a Reduced Instruction Set Computing (RISC) Processor, a Complex Instruction Set Computing (CISC) Processor, a Graphics Processing Unit (GPU), a Digital Signal Processor
10 (DSP), an Application Specific Integrated Circuit (ASIC), a Radio-Frequency Integrated Circuit (RFIC), another processor, or any suitable combination thereof) may include, for example, a processor 1312 and a processor 1314 that execute the instructions 1302. The term "processor" is intended to include multi-core processors that may comprise two or more independent processors (sometimes referred to as "cores") that may execute
15 instructions contemporaneously. Although FIG. 13 shows multiple processors 1304, the machine 1300 may include a single processor with a single-core, a single processor with multiple cores (e.g., a multi-core processor), multiple processors with a single core, multiple processors with multiples cores, or any combination thereof.

[0267] The memory 1306 includes a main memory 1316, a static memory 1318, and a
20 storage unit 1320, both accessible to the processors 1304 via the bus 1310. The main memory 1306, the static memory 1318, and storage unit 1320 store the instructions 1302 embodying any one or more of the methodologies or functions described herein. The instructions 1302 may also reside, completely or partially, within the main memory 1316, within the static memory 1318, within machine-readable medium 1322 within the storage
25 unit 1320, within at least one of the processors 1304 (e.g., within the processor's cache memory), or any suitable combination thereof, during execution thereof by the machine 1300.

[0268] The I/O components 1308 may include a wide variety of components to receive
input, provide output, produce output, transmit information, exchange information, capture
30 measurements, and so on. The specific I/O components 1308 that are included in a particular machine will depend on the type of machine. For example, portable machines such as mobile phones may include a touch input device or other such input mechanisms, while a headless server machine will likely not include such a touch input device. It will be appreciated that the I/O components 1308 may include many other components that are not
35 shown in FIG. 13. In various examples, the I/O components 1308 may include user output

components 1324 and user input components 1326. The user output components 1324 may include visual components (e.g., a display such as a plasma display panel (PDP), a light-emitting diode (LED) display, a liquid crystal display (LCD), a projector, or a cathode ray tube (CRT)), acoustic components (e.g., speakers), haptic components (e.g., a vibratory motor, resistance mechanisms), other signal generators, and so forth. The user input components 1326 may include alphanumeric input components (e.g., a keyboard, a touch screen configured to receive alphanumeric input, a photo-optical keyboard, or other alphanumeric input components), point-based input components (e.g., a mouse, a touchpad, a trackball, a joystick, a motion sensor, or another pointing instrument), tactile input components (e.g., a physical button, a touch screen that provides location and force of touches or touch gestures, or other tactile input components), audio input components (e.g., a microphone), and the like.

[0269] In further examples, the I/O components 1308 may include biometric components 1328, motion components 1330, environmental components 1332, or position components 1334, among a wide array of other components. For example, the biometric components 1328 include components to detect expressions (e.g., hand expressions, facial expressions, vocal expressions, body gestures, or eye-tracking), measure biosignals (e.g., blood pressure, heart rate, body temperature, perspiration, or brain waves), identify a person (e.g., voice identification, retinal identification, facial identification, fingerprint identification, or electroencephalogram-based identification), and the like.

[0270] The motion components 1330 include acceleration sensor components (e.g., accelerometer), gravitation sensor components, rotation sensor components (e.g., gyroscope).

[0271] The environmental components 1332 include, for example, one or more cameras (with still image/photograph and video capabilities), illumination sensor components (e.g., photometer), temperature sensor components (e.g., one or more thermometers that detect ambient temperature), humidity sensor components, pressure sensor components (e.g., barometer), acoustic sensor components (e.g., one or more microphones that detect background noise), proximity sensor components (e.g., infrared sensors that detect nearby objects), gas sensors (e.g., gas detection sensors to detect concentrations of hazardous gasses for safety or to measure pollutants in the atmosphere), or other components that may provide indications, measurements, or signals corresponding to a surrounding physical environment.

[0272] With respect to cameras, the user system 102 may have a camera system comprising, for example, front cameras on a front surface of the user system 102 and rear

cameras on a rear surface of the user system 102. The front cameras may, for example, be used to capture still images and video of a user of the user system 102 (e.g., "selfies"), which may then be augmented with augmentation data (e.g., filters) described above. The rear cameras may, for example, be used to capture still images and videos in a more traditional camera mode, with these images similarly being augmented with augmentation data. In addition to front and rear cameras, the user system 102 may also include a 360° camera for capturing 360° photographs and videos.

[0273] Further, the camera system of the user system 102 may include dual rear cameras (e.g., a primary camera as well as a depth-sensing camera), or even triple, quad or penta rear camera configurations on the front and rear sides of the user system 102. These multiple cameras systems may include a wide camera, an ultra-wide camera, a telephoto camera, a macro camera, and a depth sensor, for example.

[0274] The position components 1334 include location sensor components (e.g., a GPS receiver component), altitude sensor components (e.g., altimeters or barometers that detect air pressure from which altitude may be derived), orientation sensor components (e.g., magnetometers), and the like.

[0275] Communication may be implemented using a wide variety of technologies. The I/O components 1308 further include communication components 1336 operable to couple the machine 1300 to a network 1338 or devices 1340 via respective coupling or

connections. For example, the communication components 1336 may include a network interface component or another suitable device to interface with the network 1338. In further examples, the communication components 1336 may include wired communication components, wireless communication components, cellular communication components, Near Field Communication (NFC) components, Bluetooth® components (e.g., Bluetooth® Low Energy), Wi-Fi® components, and other communication components to provide communication via other modalities. The devices 1340 may be another machine or any of a wide variety of peripheral devices (e.g., a peripheral device coupled via a USB).

[0276] Moreover, the communication components 1336 may detect identifiers or include components operable to detect identifiers. For example, the communication components 1336 may include Radio Frequency Identification (RFID) tag reader components, NFC smart tag detection components, optical reader components (e.g., an optical sensor to detect one-dimensional bar codes such as Universal Product Code (UPC) bar code, multi-dimensional bar codes such as Quick Response (QR) code, Aztec code, Data Matrix, Dataglyph™, MaxiCode, PDF417, Ultra Code, UCC RSS-2D bar code, and other optical codes), or acoustic detection components (e.g., microphones to identify tagged audio

signals). In addition, a variety of information may be derived via the communication components 1336, such as location via Internet Protocol (IP) geolocation, location via Wi-Fi® signal triangulation, location via detecting an NFC beacon signal that may indicate a particular location, and so forth.

5 [0277] The various memories (e.g., main memory 1316, static memory 1318, and memory of the processors 1304) and storage unit 1320 may store one or more sets of instructions and data structures (e.g., software) embodying or used by any one or more of the methodologies or functions described herein. These instructions (e.g., the instructions 1302), when executed by processors 1304, cause various operations to implement the disclosed examples.

10 [0278] The instructions 1302 may be transmitted or received over the network 1338, using a transmission medium, via a network interface device (e.g., a network interface component included in the communication components 1336) and using any one of several well-known transfer protocols (e.g., hypertext transfer protocol (HTTP)). Similarly, the instructions 1302 may be transmitted or received using a transmission medium via a coupling (e.g., a
15 peer-to-peer coupling) to the devices 1340.

[0279] SOFTWARE ARCHITECTURE

[0280] FIG. 14 is a block diagram 1400 illustrating a software architecture 1402, which can be installed on any one or more of the devices described herein. The software architecture 1402 is supported by hardware such as a machine 1404 that includes processors
20 1406, memory 1408, and I/O components 1410. In this example, the software architecture 1402 can be conceptualized as a stack of layers, where each layer provides a particular functionality. The software architecture 1402 includes layers such as an operating system 1412, libraries 1414, frameworks 1416, and applications 1418. Operationally, the applications 1418 invoke API calls 1420 through the software stack and receive messages
25 1422 in response to the API calls 1420.

[0281] The operating system 1412 manages hardware resources and provides common services. The operating system 1412 includes, for example, a kernel 1424, services 1426, and drivers 1428. The kernel 1424 acts as an abstraction layer between the hardware and the other software layers. For example, the kernel 1424 provides memory management,
30 processor management (e.g., scheduling), component management, networking, and security settings, among other functionalities. The services 1426 can provide other common services for the other software layers. The drivers 1428 are responsible for controlling or interfacing with the underlying hardware. For instance, the drivers 1428 can include display drivers, camera drivers, BLUETOOTH® or BLUETOOTH® Low Energy drivers, flash memory

drivers, serial communication drivers (e.g., USB drivers), WI-FI® drivers, audio drivers, power management drivers, and so forth.

[0282] The libraries 1414 provide a common low-level infrastructure used by the applications 1418. The libraries 1414 can include system libraries 1430 (e.g., C standard library) that provide functions such as memory allocation functions, string manipulation functions, mathematic functions, and the like. In addition, the libraries 1414 can include API libraries 1432 such as media libraries (e.g., libraries to support presentation and manipulation of various media formats such as Moving Picture Experts Group-4 (MPEG4), Advanced Video Coding (H.264 or AVC), Moving Picture Experts Group Layer-3 (MP3), Advanced Audio Coding (AAC), Adaptive Multi-Rate (AMR) audio codec, Joint Photographic Experts Group (JPEG or JPG), or Portable Network Graphics (PNG)), graphics libraries (e.g., an OpenGL framework used to render in two dimensions (2D) and three dimensions (3D) in a graphic content on a display), database libraries (e.g., SQLite to provide various relational database functions), web libraries (e.g., WebKit to provide web browsing functionality), and the like. The libraries 1414 can also include a wide variety of other libraries 1434 to provide many other APIs to the applications 1418.

[0283] The frameworks 1416 provide a common high-level infrastructure that is used by the applications 1418. For example, the frameworks 1416 provide various graphical user interface (GUI) functions, high-level resource management, and high-level location services. The frameworks 1416 can provide a broad spectrum of other APIs that can be used by the applications 1418, some of which may be specific to a particular operating system or platform.

[0284] In an example, the applications 1418 may include a home application 1436, a contacts application 1438, a browser application 1440, a book reader application 1442, a location application 1444, a media application 1446, a messaging application 1448, a game application 1450, and a broad assortment of other applications such as a third-party application 1452. The applications 1418 are programs that execute functions defined in the programs. Various programming languages can be employed to create one or more of the applications 1418, structured in a variety of manners, such as object-oriented programming languages (e.g., Objective-C, Java, or C++) or procedural programming languages (e.g., C or assembly language). In a specific example, the third-party application 1452 (e.g., an application developed using the ANDROID™ or IOS™ software development kit (SDK) by an entity other than the vendor of the particular platform) may be mobile software running on a mobile operating system such as IOS™, ANDROID™, WINDOWS® Phone, or another mobile operating system. In this example, the third-party application 1452 can

invoke the API calls 1420 provided by the operating system 1412 to facilitate functionalities described herein.

MACHINE-LEARNING PIPELINE

[0285] FIG. 16 is a flowchart depicting a machine-learning pipeline 1600, according to some examples. The machine-learning pipelines 1600 may be used to generate a trained model, for example the trained machine-learning program 1602 of FIG. 16, described herein to perform operations associated with searches and query responses.

Overview

[0286] Broadly, machine learning may involve using computer algorithms to automatically learn patterns and relationships in data, potentially without the need for explicit programming to do so after the algorithm is trained. Examples of machine learning algorithms can be divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning.

- Supervised learning involves training a model using labeled data to predict an output for new, unseen inputs. Examples of supervised learning algorithms include linear regression, decision trees, and neural networks.
- Unsupervised learning involves training a model on unlabeled data to find hidden patterns and relationships in the data. Examples of unsupervised learning algorithms include clustering, principal component analysis, and generative models like autoencoders.
- Reinforcement learning involves training a model to make decisions in a dynamic environment by receiving feedback in the form of rewards or penalties. Examples of reinforcement learning algorithms include Q-learning and policy gradient methods.

[0287] Examples of specific machine learning algorithms that may be deployed, according to some examples, include logistic regression, which is a type of supervised learning algorithm used for binary classification tasks. Logistic regression models the probability of a binary response variable based on one or more predictor variables. Another example type of machine learning algorithm is Naïve Bayes, which is another supervised learning algorithm used for classification tasks. Naïve Bayes is based on Bayes' theorem and assumes that the predictor variables are independent of each other. Random Forest is another type of supervised learning algorithm used for classification, regression, and other tasks. Random Forest builds a collection of decision trees and combines their outputs to make predictions. Further examples include neural networks which consist of interconnected layers of nodes (or neurons) that process information and make predictions based on the input data. Matrix factorization is another type of machine learning algorithm used for

recommender systems and other tasks. Matrix factorization decomposes a matrix into two or more matrices to uncover hidden patterns or relationships in the data. Support Vector Machines (SVM) are a type of supervised learning algorithm used for classification, regression, and other tasks. SVM finds a hyperplane that separates the different classes in the data. Other types of machine learning algorithms include decision trees, k-nearest neighbors, clustering algorithms, and deep learning algorithms such as convolutional neural networks (CNN), recurrent neural networks (RNN), and transformer models. The choice of algorithm depends on the nature of the data, the complexity of the problem, and the performance requirements of the application.

[0288] The performance of machine learning models is typically evaluated on a separate test set of data that was not used during training to ensure that the model can generalize to new, unseen data. Evaluating the model on a separate test set helps to mitigate the risk of overfitting, a common issue in machine learning where a model learns to perform exceptionally well on the training data but fails to maintain that performance on data it hasn't encountered before. By using a test set, the system obtains a more reliable estimate of the model's real-world performance and its potential effectiveness when deployed in practical applications.

[0289] Although several specific examples of machine learning algorithms are discussed herein, the principles discussed herein can be applied to other machine learning algorithms as well. Deep learning algorithms such as convolutional neural networks, recurrent neural networks, and transformers, as well as more traditional machine learning algorithms like decision trees, random forests, and gradient boosting may be used in various machine learning applications.

[0290] Two example types of problems in machine learning are classification problems and regression problems. Classification problems, also referred to as categorization problems, aim at classifying items into one of several category values (for example, is this object an apple or an orange?). Regression algorithms aim at quantifying some items (for example, by providing a value that is a real number).

Phases

Generating a trained machine-learning program 1602 may include multiple types of phases that form part of the machine-learning pipeline 1600, including for example the following phases 1500 illustrated in FIG. 15:

- Data collection and preprocessing 1502: This may include acquiring and cleaning data to ensure that it is suitable for use in the machine learning model. Data can be gathered from user content creation and labeled using a machine learning algorithm

trained to label data. Data can be generated by applying a machine learning algorithm to identify or generate similar data. This may also include removing duplicates, handling missing values, and converting data into a suitable format.

- Feature engineering 1504: This may include selecting and transforming the training data 1604 to create features that are useful for predicting the target variable. Feature engineering may include (1) receiving features 1606 (e.g., as structured or labeled data in supervised learning) and/or (2) identifying features 1606 (e.g., unstructured or unlabeled data for unsupervised learning) in training data 1604.
- Model selection and training 1506: This may include specifying a particular problem or desired response from input data, selecting an appropriate machine learning algorithm, and training it on the preprocessed data. This may further involve splitting the data into training and testing sets, using cross-validation to evaluate the model, and tuning hyperparameters to improve performance. Model selection can be based on factors such as the type of data, problem complexity, computational resources, or desired performance.
- Model evaluation 1508: This may include evaluating the performance of a trained model (e.g., the trained machine-learning program 1602) on a separate testing dataset. This can help determine if the model is overfitting or underfitting and if it is suitable for deployment.
- Prediction 1510: This involves using a trained model (e.g., trained machine-learning program 1602) to generate predictions on new, unseen data.
- Validation, refinement or retraining 1512: This may include updating a model based on feedback generated from the prediction phase, such as new data or user feedback.
- Deployment 1514: This may include integrating the trained model (e.g., the trained machine-learning program 1602) into a larger system or application, such as a web service, mobile app, or IoT device. This can involve setting up APIs, building a user interface, and ensuring that the model is scalable and can handle large volumes of data.

[0291] FIG. 16 illustrates two example phases, namely a training phase 1608 (part of the model selection and trainings 1506) and a prediction phase 1610 (part of prediction 1510).

Prior to the training phase 1608, feature engineering 1504 is used to identify features 1606. This may include identifying informative, discriminating, and independent features for the effective operation of the trained machine-learning program 1602 in pattern recognition, classification, and regression. In some examples, the training data 1604 includes labeled data, which is known data for pre-identified features 1606 and one or more outcomes.

[0292] Each of the features 1606 may be a variable or attribute, such as individual measurable property of a process, article, system, or phenomenon represented by a data set (e.g., the training data 1604). Features 1606 may also be of different types, such as numeric features, strings, vectors, matrices, encodings, and graphs, and may include one or more of content 1612, concepts 1614, attributes 1616, historical data 1618 and/or user data 1620, merely for example. Concept features can include abstract relationships or patterns in data, such as determining a topic of a document or discussion in a chat window between users. Content features include determining a context based on input information, such as determining a context of a user based on user interactions or surrounding environmental factors. Context features can include text features, such as frequency or preference of words or phrases, image features, such as pixels, textures, or pattern recognition, audio classification, such as spectrograms, and/or the like. Attribute features include intrinsic attributes (directly observable) or extrinsic features (derived), such as identifying square footage, location, or age of a real estate property identified in a camera feed. User data features include data pertaining to a particular individual or to a group of individuals, such as in a geographical location or that share demographic characteristics. User data can include demographic data (such as age, gender, location, or occupation), user behavior (such as browsing history, purchase history, conversion rates, click-through rates, or engagement metrics), or user preferences (such as preferences to certain video, text, or digital content items). Historical data includes past events or trends that can help identify patterns or relationships over time.

[0293] In training phases 1608, the machine-learning pipeline 1600 uses the training data 1604 to find correlations among the features 1606 that affect a predicted outcome or prediction/inference data 1622.

[0294] With the training data 1604 and the identified features 1606, the trained machine-learning program 1602 is trained during the training phase 1608 during machine-learning program training 1624. The machine-learning program training 1624 appraises values of the features 1606 as they correlate to the training data 1604. The result of the training is the trained machine-learning program 1602 (e.g., a trained or learned model).

[0295] Further, the training phase 1608 may involve machine learning, in which the training data 1604 is structured (e.g., labeled during preprocessing operations), and the trained machine-learning program 1602 implements a relatively simple neural network 1626 capable of performing, for example, classification and clustering operations. In other examples, the training phase 1608 may involve deep learning, in which the training data 1604 is unstructured, and the trained machine-learning program 1602 implements a deep

neural network 1626 that is able to perform both feature extraction and classification/clustering operations.

[0296] A neural network 1626 may, in some examples, be generated during the training phase 1608, and implemented within the trained machine-learning program 1602. The

5 neural network 1626 includes a hierarchical (e.g., layered) organization of neurons, with each layer including multiple neurons or nodes. Neurons in the input layer receive the input data, while neurons in the output layer produce the final output of the network. Between the input and output layers, there may be one or more hidden layers, each including multiple neurons.

10 **[0297]** Each neuron in the neural network 1626 operationally computes a small function, such as an activation function that takes as input the weighted sum of the outputs of the neurons in the previous layer, as well as a bias term. The output of this function is then passed as input to the neurons in the next layer. If the output of the activation function exceeds a certain threshold, an output is communicated from that neuron (e.g., transmitting
15 neuron) to a connected neuron (e.g., receiving neuron) in successive layers. The connections between neurons have associated weights, which define the influence of the input from a transmitting neuron to a receiving neuron. During the training phase, these weights are adjusted by the learning algorithm to optimize the performance of the network. Different types of neural networks may use different activation functions and learning algorithms,
20 which can affect their performance on different tasks. Overall, the layered organization of neurons and the use of activation functions and weights enable neural networks to model complex relationships between inputs and outputs, and to generalize to new inputs that were not seen during training.

[0298] In some examples, the neural network 1626 may also be one of a number of
25 different types of neural networks or a combination thereof, such as a single-layer feed-forward network, a Multilayer Perceptron (MLP), an Artificial Neural Network (ANN), a Recurrent Neural Network (RNN), a Long Short-Term Memory Network (LSTM), a Bidirectional Neural Network, a symmetrically connected neural network, a Deep Belief Network (DBN), a Convolutional Neural Network (CNN), a Generative Adversarial
30 Network (GAN), an Autoencoder Neural Network (AE), a Restricted Boltzmann Machine (RBM), a Hopfield Network, a Self-Organizing Map (SOM), a Radial Basis Function Network (RBFN), a Spiking Neural Network (SNN), a Liquid State Machine (LSM), an Echo State Network (ESN), a Neural Turing Machine (NTM), or a Transformer Network, merely for example.

[0299] In addition to the training phase 1608, a validation phase may be performed evaluated on a separate dataset known as the validation dataset. The validation dataset is used to tune the hyperparameters of a model, such as the learning rate and the regularization parameter. The hyperparameters are adjusted to improve the performance of the model on the validation dataset.

[0300] The neural network 1626 is iteratively trained by adjusting model parameters to minimize a specific loss function or maximize a certain objective. The system can continue to train the neural network 1626 by adjusting parameters based on the output of the validation, refinement, or retraining block 1512, and rerun the prediction 1510 on new or already run training data. The system can employ optimization techniques for these adjustments such as gradient descent algorithms, momentum algorithms, Nesterov Accelerated Gradient (NAG) algorithm, and/or the like. The system can continue to iteratively train the neural network 1626 even after deployment 1514 of the neural network 1626. The neural network 1626 can be continuously trained as new data emerges, such as based on user creation or system-generated training data.

[0301] Once a model is fully trained and validated, in a testing phase, the model may be tested on a new dataset that the model has not seen before. The testing dataset is used to evaluate the performance of the model and to ensure that the model has not overfit the training data.

[0302] In prediction phase 1610, the trained machine-learning program 1602 uses the features 1606 for analyzing query data 1628 to generate inferences, outcomes, or predictions, as examples of a prediction/inference data 1622. For example, during prediction phase 1610, the trained machine-learning program 1602 is used to generate an output. Query data 1628 is provided as an input to the trained machine-learning program 1602, and the trained machine-learning program 1602 generates the prediction/inference data 1622 as output, responsive to receipt of the query data 1628. Query data can include a prompt, such as a user entering a textual question or speaking a question audibly. In some cases, the system generates the query based on an interaction function occurring in the system, such as a user interacting with a virtual object, a user sending another user a question in a chat window, or an object detected in a camera feed.

[0303] In some examples the trained machine-learning program 1602 may be a generative AI model. Generative AI is a term that may refer to any type of artificial intelligence that can create new content from training data 1604. For example, generative AI can produce text, images, video, audio, code or synthetic data that are similar to the original data but not identical.

[0304] Some of the techniques that may be used in generative AI are:

- Convolutional Neural Networks (CNNs): CNNs are commonly used for image recognition and computer vision tasks. They are designed to extract features from images by using filters or kernels that scan the input image and highlight important patterns. CNNs may be used in applications such as object detection, facial recognition, and autonomous driving.
- Recurrent Neural Networks (RNNs): RNNs are designed for processing sequential data, such as speech, text, and time series data. They have feedback loops that allow them to capture temporal dependencies and remember past inputs. RNNs may be used in applications such as speech recognition, machine translation, and sentiment analysis
- Generative adversarial networks (GANs): These are models that consist of two neural networks: a generator and a discriminator. The generator tries to create realistic content that can fool the discriminator, while the discriminator tries to distinguish between real and fake content. The two networks compete with each other and improve over time. GANs may be used in applications such as image synthesis, video prediction, and style transfer.
- Variational autoencoders (VAEs): These are models that encode input data into a latent space (a compressed representation) and then decode it back into output data. The latent space can be manipulated to generate new variations of the output data. They may use self-attention mechanisms to process input data, allowing them to handle long sequences of text and capture complex dependencies.
- Transformer models: These are models that use attention mechanisms to learn the relationships between different parts of input data (such as words or pixels) and generate output data based on these relationships. Transformer models can handle sequential data such as text or speech as well as non-sequential data such as images or code.

[0305] In generative AI examples, the prediction/inference data that is output include trend assessment and predictions, translations, summaries, image or video recognition and categorization, natural language processing, face recognition, user sentiment assessments, advertisement targeting and optimization, voice recognition, or media content generation, recommendation, and personalization.

[0306] EXAMPLES

[0307] In view of the above-described implementations of subject matter this application discloses the following list of examples, wherein one feature of an example in isolation or more than one feature of an example, taken in combination and, optionally, in combination

with one or more features of one or more further examples are further examples also falling within the disclosure of this application.

[0308] Example 1 is a system comprising: at least one processor; and at least one memory component storing instructions that, when executed by the at least one processor, cause the at least one processor to perform operations comprising: identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text; identifying a number of channels in a decoder of the autoencoder, the decoder being configured to receive latent features as input and output images; identifying a first performance characteristic of the decoder; changing the node topology of the decoder based on the first performance characteristic to generate an updated decoder; and retraining the latent diffusion machine learning model using the updated decoder by: inputting latent features to the updated decoder, assessing an outputted image using the updated decoder, and updating one or more weights of the decoder based on the assessment of the outputted image.

[0309] In Example 2, the subject matter of Example 1 includes, wherein the first performance characteristic includes a magnitude of the output for a particular channel.

[0310] In Example 3, the subject matter of Example 2 includes, wherein the changing of the node topology includes determining that the magnitude for the particular channel is above a minimum threshold and determining to change the node topology for the particular channel.

[0311] In Example 4, the subject matter of Examples 1–3 includes, wherein the first performance characteristic includes a reconstruction loss that measures how well the autoencoder reconstructs the input data to image data.

[0312] In Example 5, the subject matter of Examples 1–4 includes, wherein changing the node topology includes removing one or more channels of the decoder.

[0313] In Example 6, the subject matter of Examples 1–5 includes, wherein the operations further comprise: identifying first performance characteristics for each channel in the decoder, wherein changing the node topology of the decoder comprises removing half of the channels with the lowest values for the first performance characteristics.

[0314] In Example 7, the subject matter of Examples 1–6 includes, wherein changing the node topology includes reducing channel dimensions in at least one layer of the decoder to obtain a compressed decoder, wherein the compressed decoder processes latent features with lower latency and fewer parameters than to decoder.

[0315] In Example 8, the subject matter of Examples 1–7 includes, wherein changing the node topology includes removing or adding one or more channels of the decoder until two or more performance thresholds are met.

[0316] In Example 9, the subject matter of Example 8 includes, wherein the operations include removing one or more channels in response to a first performance threshold being met, and adding one or more channels in response to a second performance threshold being met.

[0317] In Example 10, the subject matter of Examples 8–9 includes, wherein the operations further comprise repeatedly adding and removing channels of the decoder until two performance thresholds are met.

[0318] In Example 11, the subject matter of Examples 8–10 includes, wherein adding the one or more channels of the decoder includes identifying one or more existing channels of the decoder that meet a performance threshold for a second performance characteristic, and copying the one or more existing channels of the decoder to add as new channels to the decoder.

[0319] In Example 12, the subject matter of Examples 8–11 includes, wherein the first and second performance characteristics are of the same type.

[0320] In Example 13, the subject matter of Examples 8–12 includes, wherein the first and second performance characteristics are of different types.

[0321] In Example 14, the subject matter of Examples 1–13 includes, wherein assessing the outputted image using the updated decoder includes comparing the outputted image of the updated decoder with an outputted image of the original decoder, wherein the outputted image of the updated decoder and the outputted image of the original decoder is generated using the same input data.

[0322] In Example 15, the subject matter of Example 14 includes, wherein the operations further comprise inputting random noise data into the latent diffusion machine learning model, wherein the input data to the updated decoder and the original decoder includes latent features generated by the latent diffusion machine learning model from the inputted random noise data.

[0323] In Example 16, the subject matter of Examples 14–15 includes, wherein comparing the outputted image of the updated decoder with an outputted image of the original decoder includes determining a mean squared error between the outputted images, wherein the operations further comprise: updating one or more weights of the updated decoder; and repeatedly inputting of the random noise data, comparing the outputted image of the updated

decoder with an outputted image of the original decoder, and updating the weights of the updated decoder to reduce the mean squared error between the outputted images.

[0324] In Example 17, the subject matter of Examples 1–16 includes, wherein the outputted image is a frame of a video, wherein the operations are repeated to generate other frames for the video.

[0325] In Example 18, the subject matter of Examples 1–17 includes, wherein the operations further comprise: generating a prompt based on user interaction data; inputting the prompt into the latent diffusion machine learning model causing the output of the latent diffusion machine learning model to input latent features into the updated decoder with the updated weights; and receiving an image generated by the updated decoder.

[0326] Example 19 is a method comprising: identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text; identifying a number of channels in a decoder of the autoencoder, the decoder being configured to receive latent features as input and output images; identifying a performance characteristic of the decoder; changing the node topology of the decoder based on the performance characteristic to generate an updated decoder; and retraining the latent diffusion machine learning model using the updated decoder by: inputting latent features to the updated decoder, assessing an outputted image using the updated decoder, and updating one or more weights of the decoder based on the assessment of the outputted image.

[0327] Example 20 is a non-transitory computer-readable storage medium, the computer-readable storage medium including instructions that when executed by a computer, cause the computer to: identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text; identifying a number of channels in a decoder of the autoencoder, the decoder being configured to receive latent features as input and output images; identifying a performance characteristic of the decoder; changing the node topology of the decoder based on the performance characteristic to generate an updated decoder; and retraining the latent diffusion machine learning model using the updated decoder by: inputting latent features to the updated decoder, assessing an outputted image using the updated decoder, and updating one or more weights of the decoder based on the assessment of the outputted image.

[0328] Example 21 is at least one machine-readable medium including instructions that, when executed by processing circuitry, cause the processing circuitry to perform operations to implement Examples 1–20.

[0329] Example 22 is an apparatus comprising means to implement Examples 1–20.

[0330] Example 23 is a system to implement Examples 1–20.

[0331] Example 24 is a method to implement Examples 1–20.

[0332] **GLOSSARY**

5 [0333] "Client device" refers, for example, to any machine that interfaces to a communications network to obtain resources from one or more server systems or other client devices. A client device may be, but is not limited to, a mobile phone, desktop computer, laptop, portable digital assistants (PDAs), smartphones, tablets, ultrabooks, netbooks, laptops, multi-processor systems, microprocessor-based or programmable
10 consumer electronics, game consoles, set-top boxes, or any other communication device that a user may use to access a network.

[0334] "Communication network" refers, for example, to one or more portions of a network that may be an ad hoc network, an intranet, an extranet, a virtual private network (VPN), a local area network (LAN), a wireless LAN (WLAN), a wide area network (WAN),
15 a wireless WAN (WWAN), a metropolitan area network (MAN), the Internet, a portion of the Internet, a portion of the Public Switched Telephone Network (PSTN), a plain old telephone service (POTS) network, a cellular telephone network, a wireless network, a Wi-Fi® network, another type of network, or a combination of two or more such networks. For example, a network or a portion of a network may include a wireless or cellular network,
20 and the coupling may be a Code Division Multiple Access (CDMA) connection, a Global System for Mobile communications (GSM) connection, or other types of cellular or wireless coupling. In this example, the coupling may implement any of a variety of types of data transfer technology, such as Single Carrier Radio Transmission Technology (1xRTT), Evolution-Data Optimized (EVDO) technology, General Packet Radio Service (GPRS)
25 technology, Enhanced Data rates for GSM Evolution (EDGE) technology, third Generation Partnership Project (3GPP) including 3G, fourth-generation wireless (4G) networks, Universal Mobile Telecommunications System (UMTS), High Speed Packet Access (HSPA), Worldwide Interoperability for Microwave Access (WiMAX), Long Term Evolution (LTE) standard, others defined by various standard-setting organizations, other
30 long-range protocols, or other data transfer technology.

[0335] "Component" refers, for example, to a device, physical entity, or logic having boundaries defined by function or subroutine calls, branch points, APIs, or other technologies that provide for the partitioning or modularization of particular processing or control functions. Components may be combined via their interfaces with other components
35 to carry out a machine process. A component may be a packaged functional hardware unit

designed for use with other components and a part of a program that usually performs a particular function of related functions. Components may constitute either software components (e.g., code embodied on a machine-readable medium) or hardware components. A "hardware component" is a tangible unit capable of performing certain operations and

5 may be configured or arranged in a certain physical manner. In various examples, one or more computer systems (e.g., a standalone computer system, a client computer system, or a server computer system) or one or more hardware components of a computer system (e.g., a processor or a group of processors) may be configured by software (e.g., an application or application portion) as a hardware component that operates to perform certain operations as

10 described herein. A hardware component may also be implemented mechanically, electronically, or any suitable combination thereof. For example, a hardware component may include dedicated circuitry or logic that is permanently configured to perform certain operations. A hardware component may be a special-purpose processor, such as a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC). A

15 hardware component may also include programmable logic or circuitry that is temporarily configured by software to perform certain operations. For example, a hardware component may include software executed by a general-purpose processor or other programmable processors. Once configured by such software, hardware components become specific machines (or specific components of a machine) uniquely tailored to perform the configured

20 functions and are no longer general-purpose processors. It will be appreciated that the decision to implement a hardware component mechanically, in dedicated and permanently configured circuitry, or in temporarily configured circuitry (e.g., configured by software), may be driven by cost and time considerations. Accordingly, the phrase "hardware component"(or "hardware-implemented component") should be understood to encompass a

25 tangible entity, be that an entity that is physically constructed, permanently configured (e.g., hardwired), or temporarily configured (e.g., programmed) to operate in a certain manner or to perform certain operations described herein. Considering examples in which hardware components are temporarily configured (e.g., programmed), each of the hardware components need not be configured or instantiated at any one instance in time. For example,

30 where a hardware component comprises a general-purpose processor configured by software to become a special-purpose processor, the general-purpose processor may be configured as respectively different special-purpose processors (e.g., comprising different hardware components) at different times. Software accordingly configures a particular processor or processors, for example, to constitute a particular hardware component at one instance of

35 time and to constitute a different hardware component at a different instance of time.

Hardware components can provide information to, and receive information from, other hardware components. Accordingly, the described hardware components may be regarded as being communicatively coupled. Where multiple hardware components exist contemporaneously, communications may be achieved through signal transmission (e.g., over appropriate circuits and buses) between or among two or more of the hardware components. In examples in which multiple hardware components are configured or instantiated at different times, communications between such hardware components may be achieved, for example, through the storage and retrieval of information in memory structures to which the multiple hardware components have access. For example, one hardware component may perform an operation and store the output of that operation in a memory device to which it is communicatively coupled. A further hardware component may then, at a later time, access the memory device to retrieve and process the stored output. Hardware components may also initiate communications with input or output devices, and can operate on a resource (e.g., a collection of information). The various operations of example methods described herein may be performed, at least partially, by one or more processors that are temporarily configured (e.g., by software) or permanently configured to perform the relevant operations. Whether temporarily or permanently configured, such processors may constitute processor-implemented components that operate to perform one or more operations or functions described herein. As used herein, "processor-implemented component" refers to a hardware component implemented using one or more processors. Similarly, the methods described herein may be at least partially processor-implemented, with a particular processor or processors being an example of hardware. For example, at least some of the operations of a method may be performed by one or more processors or processor-implemented components. Moreover, the one or more processors may also operate to support performance of the relevant operations in a "cloud computing" environment or as a "software as a service" (SaaS). For example, at least some of the operations may be performed by a group of computers (as examples of machines including processors), with these operations being accessible via a network (e.g., the Internet) and via one or more appropriate interfaces (e.g., an API). The performance of certain of the operations may be distributed among the processors, not only residing within a single machine, but deployed across a number of machines. In some examples, the processors or processor-implemented components may be located in a single geographic location (e.g., within a home environment, an office environment, or a server farm). In other examples, the processors or processor-implemented components may be distributed across a number of geographic locations.

[0336] "Computer-readable storage medium" refers, for example, to both machine-storage media and transmission media. Thus, the terms include both storage devices/media and carrier waves/modulated data signals. The terms "machine-readable medium," "computer-readable medium" and "device-readable medium" mean the same thing and may be used interchangeably in this disclosure.

[0337] "Machine storage medium" refers, for example, to a single or multiple storage devices and media (e.g., a centralized or distributed database, and associated caches and servers) that store executable instructions, routines and data. The term shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media, including memory internal or external to processors. Specific examples of machine-storage media, computer-storage media and device-storage media include non-volatile memory, including by way of example semiconductor memory devices, e.g., erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), FPGA, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The terms "machine-storage medium," "device-storage medium," "computer-storage medium" mean the same thing and may be used interchangeably in this disclosure. The terms "machine-storage media," "computer-storage media," and "device-storage media" specifically exclude carrier waves, modulated data signals, and other such media, at least some of which are covered under the term "signal medium."

[0338] "Non-transitory computer-readable storage medium" refers, for example, to a tangible medium that is capable of storing, encoding, or carrying the instructions for execution by a machine.

CONCLUSION

[0339] As used in this disclosure, phrases of the form "at least one of an A, a B, or a C," "at least one of A, B, or C," "at least one of A, B, and C," and the like, should be interpreted to select at least one from the group that comprises "A, B, and C." Unless explicitly stated otherwise in connection with a particular instance in this disclosure, this manner of phrasing does not mean "at least one of A, at least one of B, and at least one of C." As used in this disclosure, the example "at least one of an A, a B, or a C," would cover any of the following selections: {A}, {B}, {C}, {A, B}, {A, C}, {B, C}, and {A, B, C}.

[0340] Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense, i.e., in the sense of "including, but not limited to." As used herein, the terms "connected," "coupled," or any variant thereof

means any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words “herein,” “above,” “below,” and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. Where the context permits, words using the singular or plural number may also include the plural or singular number respectively. The word “or” in reference to a list of two or more items, covers all of the following interpretations of the word: any one of the items in the list, all of the items in the list, and any combination of the items in the list. Likewise, the term “and/or” in reference to a list of two or more items, covers all of the following interpretations of the word: any one of the items in the list, all of the items in the list, and any combination of the items in the list.

[0341] Although some examples, e.g., those depicted in the drawings, include a particular sequence of operations, the sequence may be altered without departing from the scope of the present disclosure. For example, some of the operations depicted may be performed in parallel or in a different sequence that does not materially affect the functions as described in the examples. In other examples, different components of an example device or system that implements an example method may perform functions at substantially the same time or in a specific sequence.

[0342] The various features, steps, and processes described herein may be used independently of one another, or may be combined in various ways. All possible combinations and subcombinations are intended to fall within the scope of this disclosure. In addition, certain method or process blocks may be omitted in some implementations.

What is claimed is:

1. A system comprising:
 - at least one processor; and
 - at least one memory component storing instructions that, when executed by the at least one processor, cause the at least one processor to perform operations comprising:
 - identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text;
 - identifying a number of channels in a decoder of the autoencoder, the decoder configured to receive latent features as input and to output images;
 - identifying a first performance characteristic of the decoder;
 - changing a node topology of the decoder based on the first performance characteristic to generate an updated decoder; and
 - retraining the latent diffusion machine learning model using the updated decoder by performing operations comprising:
 - inputting latent features into the updated decoder;
 - receiving an outputted image from the updated decoder; and
 - updating one or more weights of the updated decoder based on assessment of the outputted image.
2. The system of claim 1, wherein the first performance characteristic includes a magnitude of the output for a particular channel.
3. The system of claim 2, wherein changing the node topology includes:
 - determining that the magnitude for the particular channel is above a minimum threshold; and
 - changing the node topology for the particular channel based on the determination that the magnitude is above the minimum threshold.
4. The system of claim 1, wherein the first performance characteristic includes a reconstruction loss that measures a loss of reconstruction by the autoencoder converting text to image data.
5. The system of claim 1, wherein changing the node topology includes removing one or more channels of the decoder.

6. The system of claim 1, the operations further comprising:
identifying the first performance characteristic for each channel in the decoder, and
wherein changing the node topology of the decoder comprises removing half of the channels with lowest values for the first performance characteristics.
7. The system of claim 1, wherein changing the node topology includes reducing channel dimensions in at least one layer of the decoder to obtain a compressed decoder, wherein the compressed decoder processes latent features with lower latency and fewer parameters than to decoder.
8. The system of claim 1, wherein changing the node topology includes removing or adding one or more channels of the decoder until two or more performance thresholds are met.
9. The system of claim 8, the operations further comprising:
removing one or more channels in response to a first performance threshold being met, and adding one or more channels in response to a second performance threshold being met.
10. The system of claim 8, the operations further comprising:
repeatedly adding and removing channels of the decoder until two performance thresholds are met.
11. The system of claim 8, wherein adding the one or more channels of the decoder includes identifying one or more existing channels of the decoder that meet a performance threshold for a second performance characteristic, and copying the one or more existing channels of the decoder to add as new channels to the decoder.
12. The system of claim 11, wherein the first performance characteristic and second performance characteristic are of a same type.
13. The system of claim 11, wherein the first performance characteristic and second performance characteristic are of different types.

14. The system of claim 1, wherein the assessment of the outputted image comprises comparing the outputted image of the updated decoder with an outputted image of the decoder, wherein the outputted image of the updated decoder and the outputted image of the decoder are generated using the same input data.

15. The system of claim 14, the operations further comprising:

generating latent features by inputting random noise data into the latent diffusion machine learning model, wherein data that is input into the updated decoder and the decoder includes the latent features, the same input data including the random noise data.

16. The system of claim 14, wherein comparing the outputted image of the updated decoder with an outputted image of the decoder includes determining a mean squared error between the outputted image of the updated decoder and the outputted image of the decoder, and the operations further comprise:

updating one or more weights of the updated decoder based on the mean squared error; and

repeatedly inputting random noise data into the updated decoder with the updated weights, comparing the outputted image of the updated decoder with an outputted image of the decoder, and updating the weights of the updated decoder until the mean squared error between the outputted images meets a mean squared error threshold.

17. The system of claim 1, wherein the outputted image is a frame of a video, wherein the operations performed by the at least one processor are repeated to generate other frames for the video.

18. The system of claim 1, the operations further comprising:

generating a prompt based on user interaction data;

inputting the prompt into a latent feature generator of the latent diffusion machine learning model causing the latent feature generator to output latent features that are inputted into the updated decoder with the updated weights; and

receiving an image generated by the updated decoder based on the latent features inputted into the updated decoder with the updated weights.

19. A method comprising:

identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text;

identifying a number of channels in a decoder of the autoencoder, the decoder configured to receive latent features as input and to output images;

identifying a first performance characteristic of the decoder;

changing a node topology of the decoder based on the first performance characteristic to generate an updated decoder; and

retraining the latent diffusion machine learning model using the updated decoder by performing operations comprising:

inputting latent features into the updated decoder;

receiving an outputted image from the updated decoder; and

updating one or more weights of the updated decoder based on assessment of the outputted image.

20. A non-transitory computer-readable storage medium, the computer-readable storage medium including instructions that when executed by a computer, cause the computer to:

identifying an autoencoder for a latent diffusion machine learning model, the latent diffusion machine learning model trained to receive text as input and output an image based on the received text;

identifying a number of channels in a decoder of the autoencoder, the decoder configured to receive latent features as input and to output images;

identifying a first performance characteristic of the decoder;

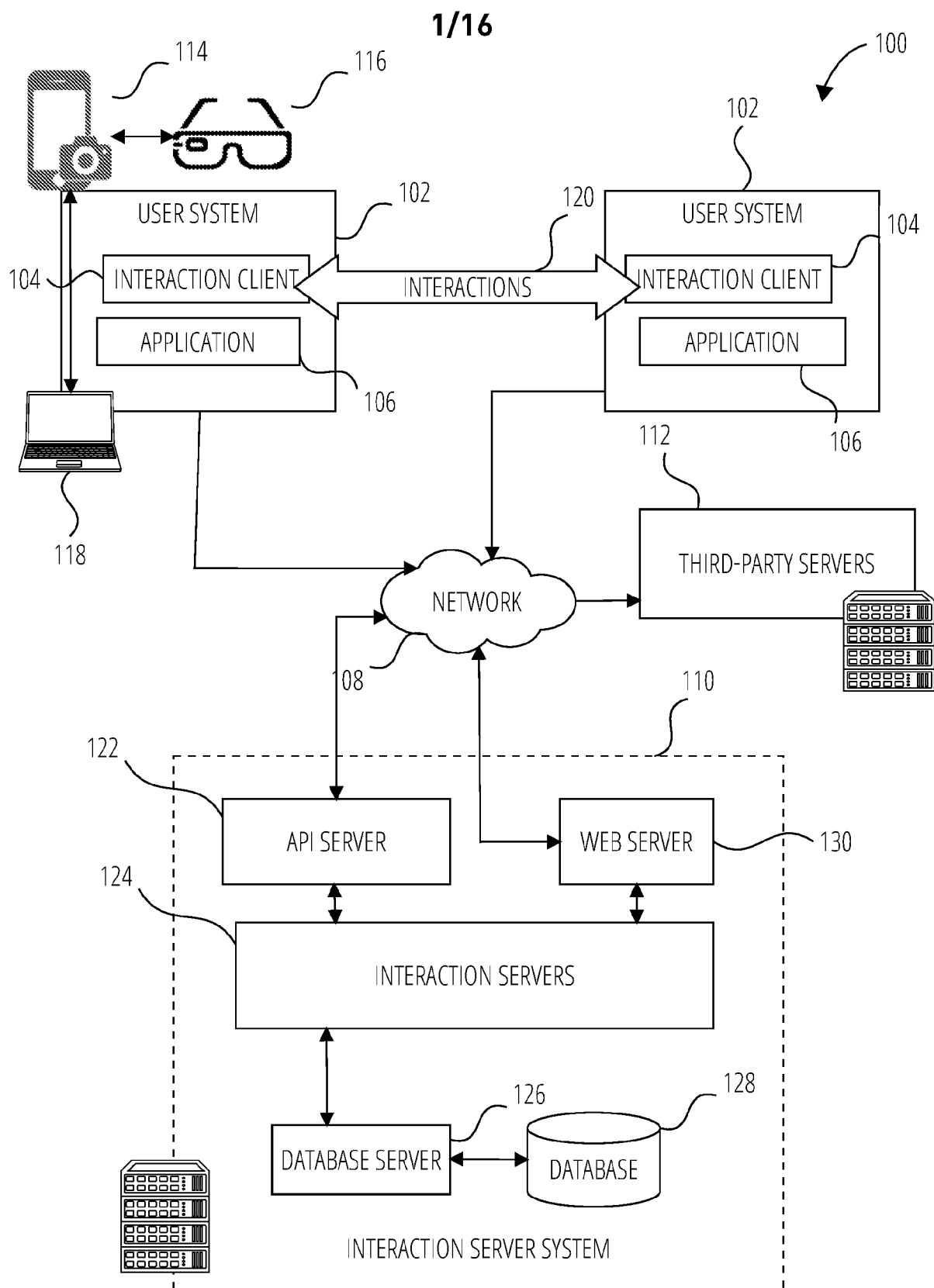
changing a node topology of the decoder based on the first performance characteristic to generate an updated decoder; and

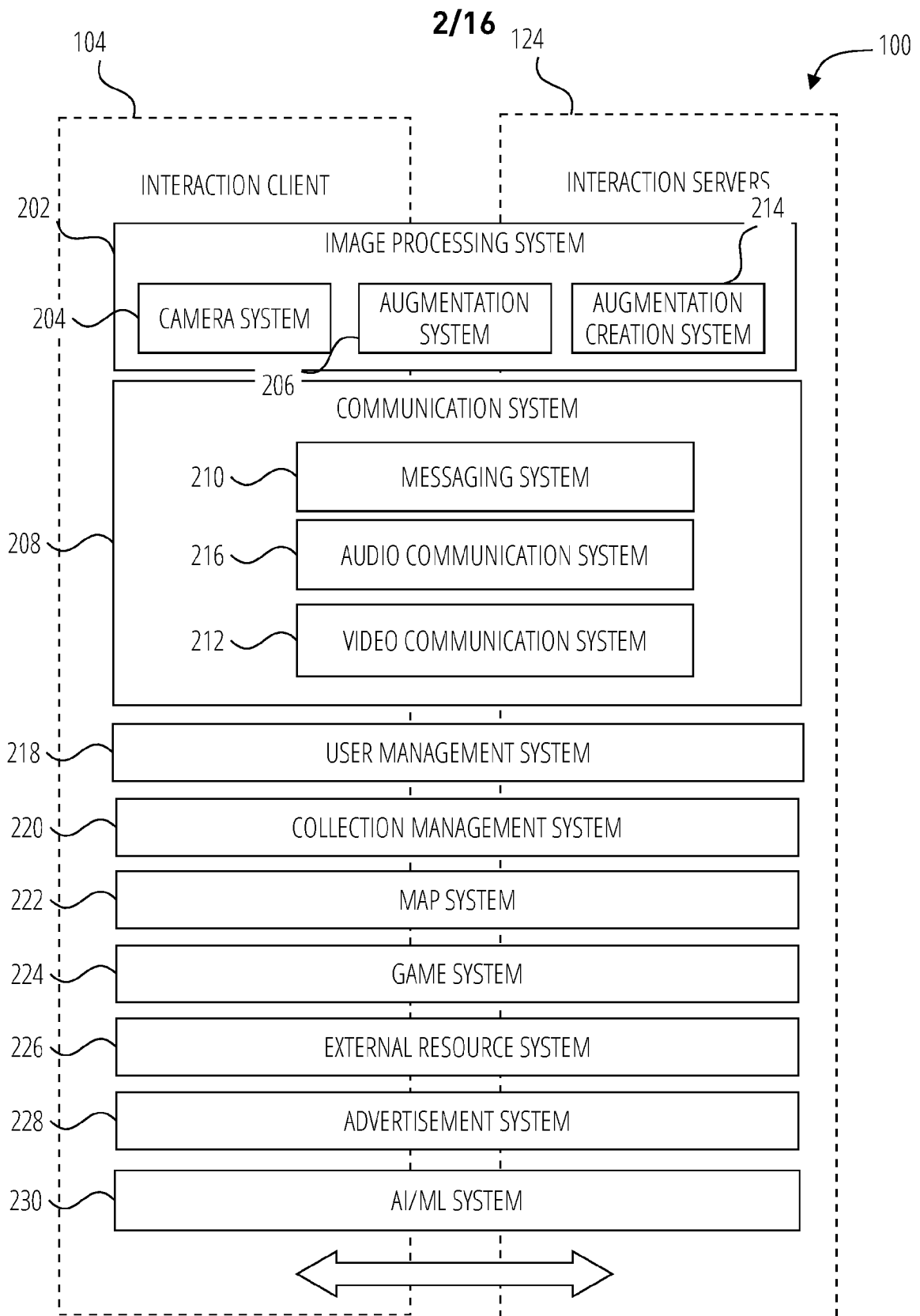
retraining the latent diffusion machine learning model using the updated decoder by performing operations comprising:

inputting latent features into the updated decoder;

receiving an outputted image from the updated decoder; and

updating one or more weights of the updated decoder based on assessment of the outputted image.

**FIG. 1**

**FIG. 2**

3/16

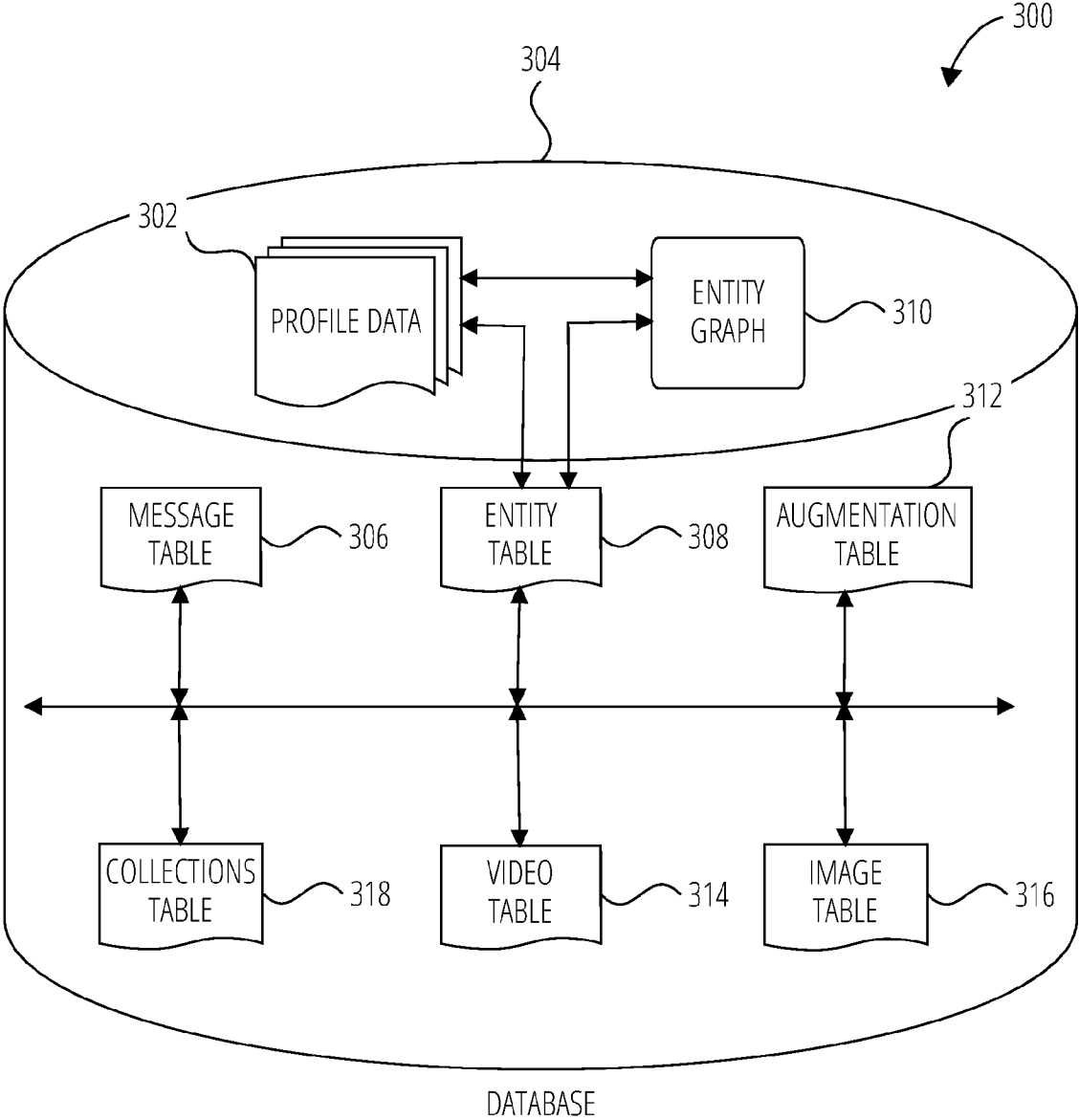


FIG. 3

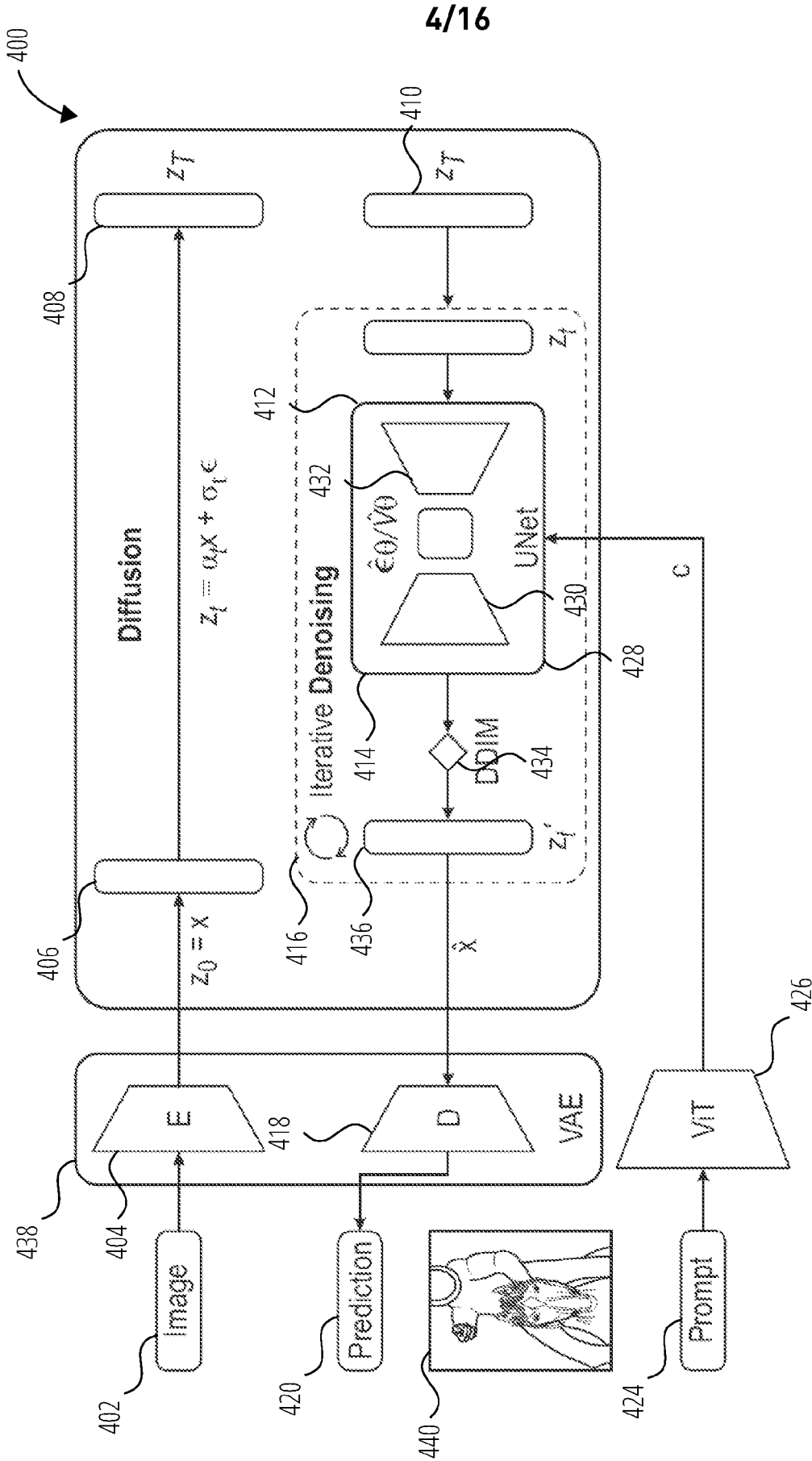


FIG. 4

5/16

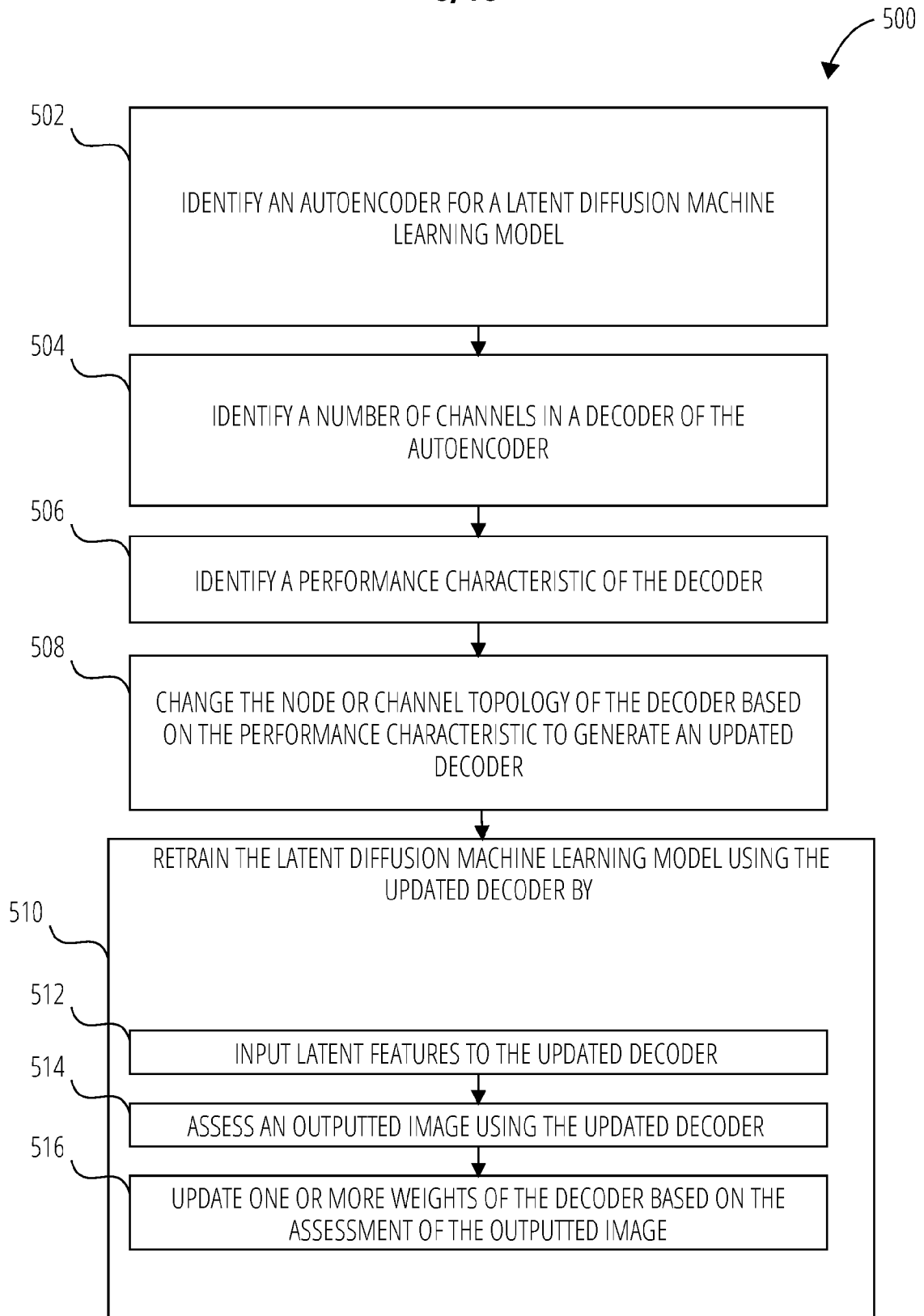


FIG. 5

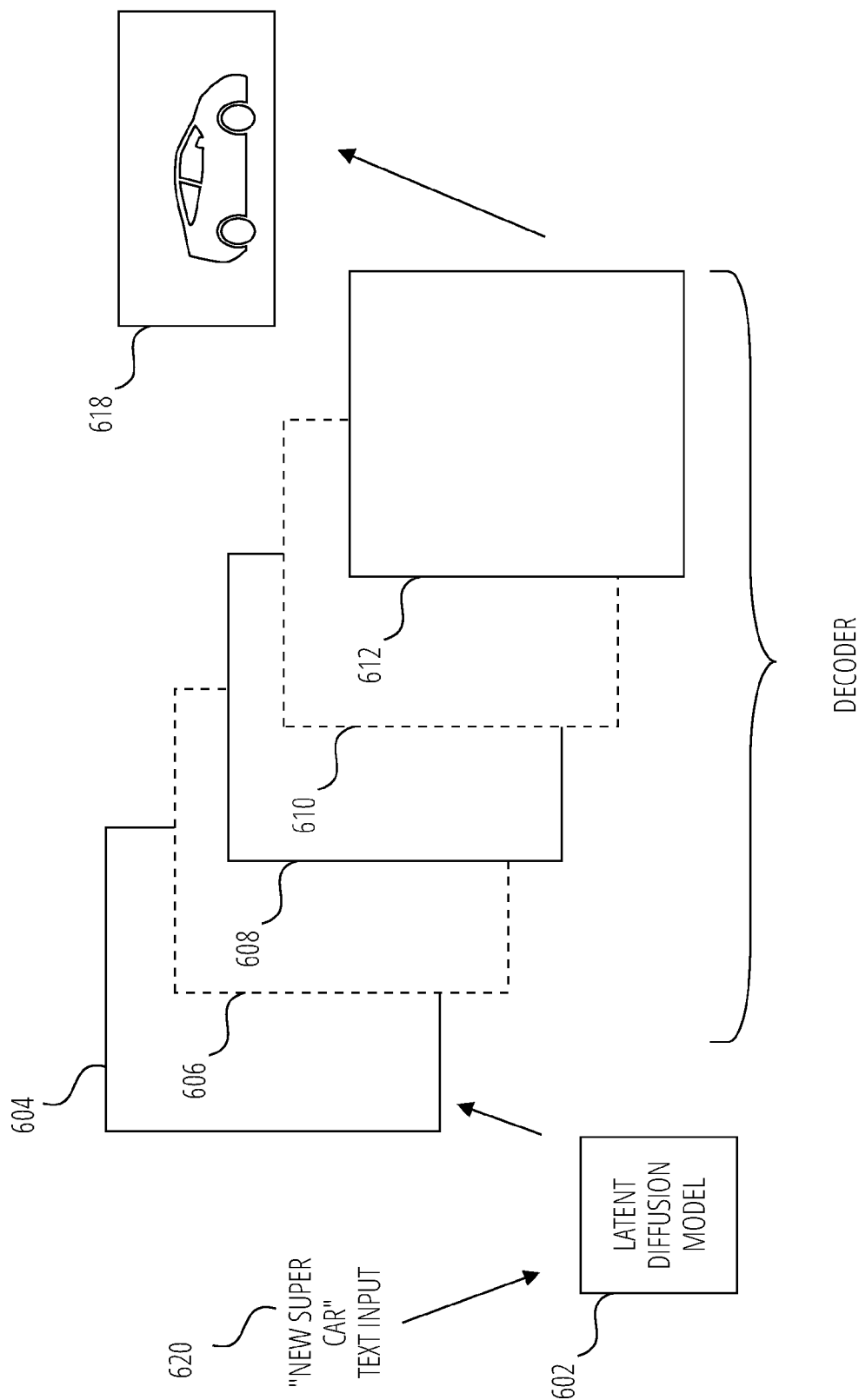
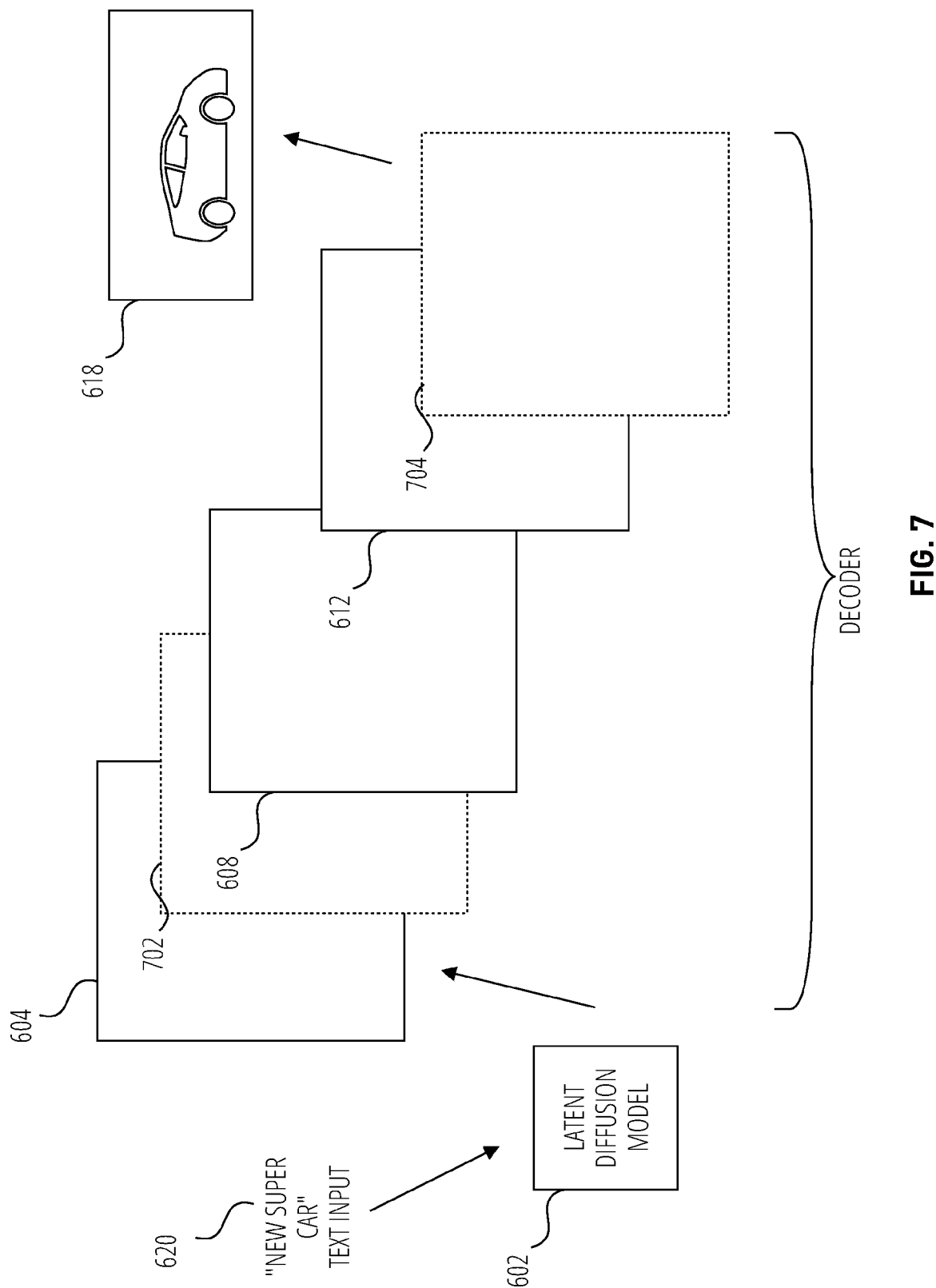
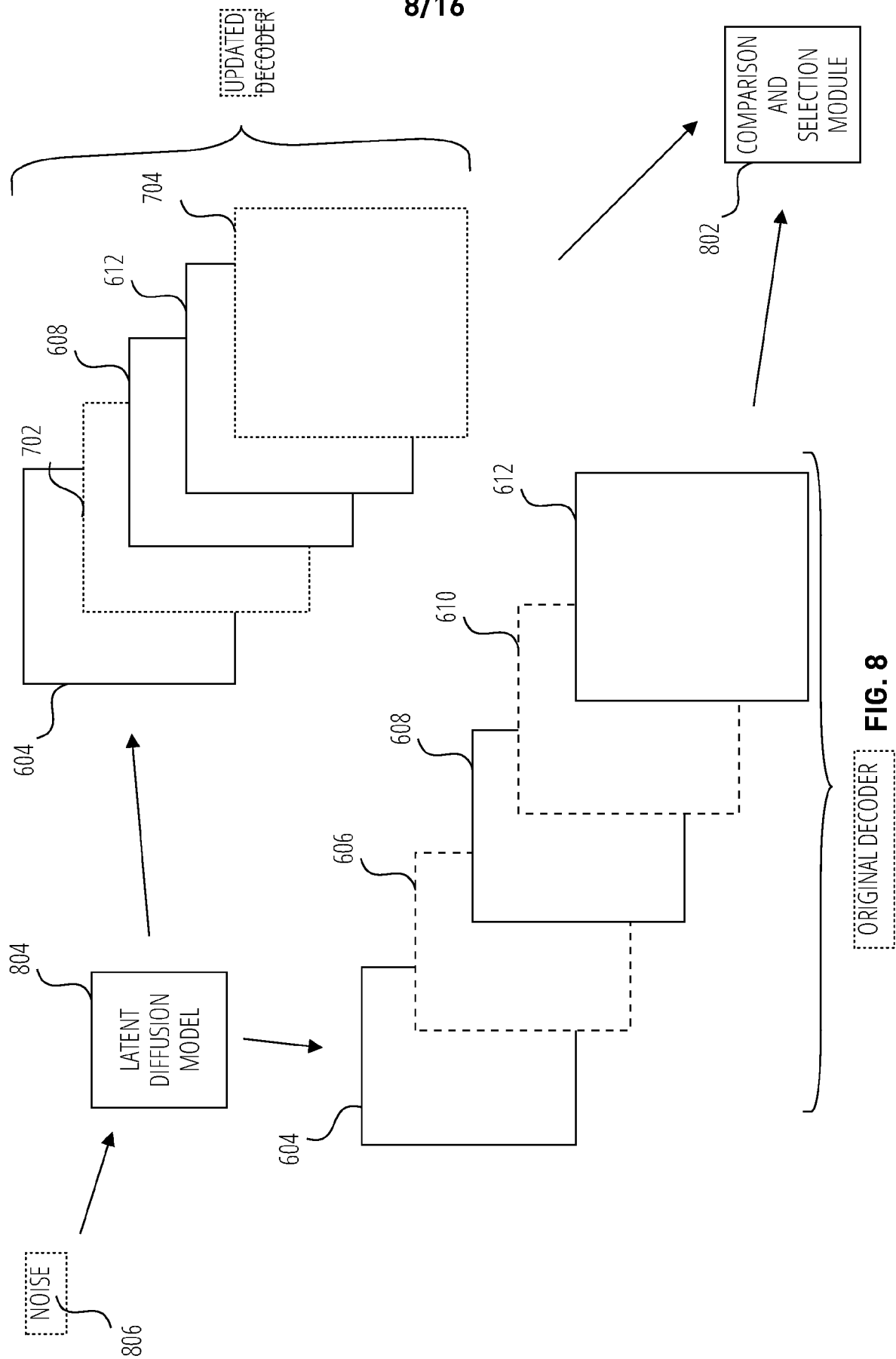


FIG. 6





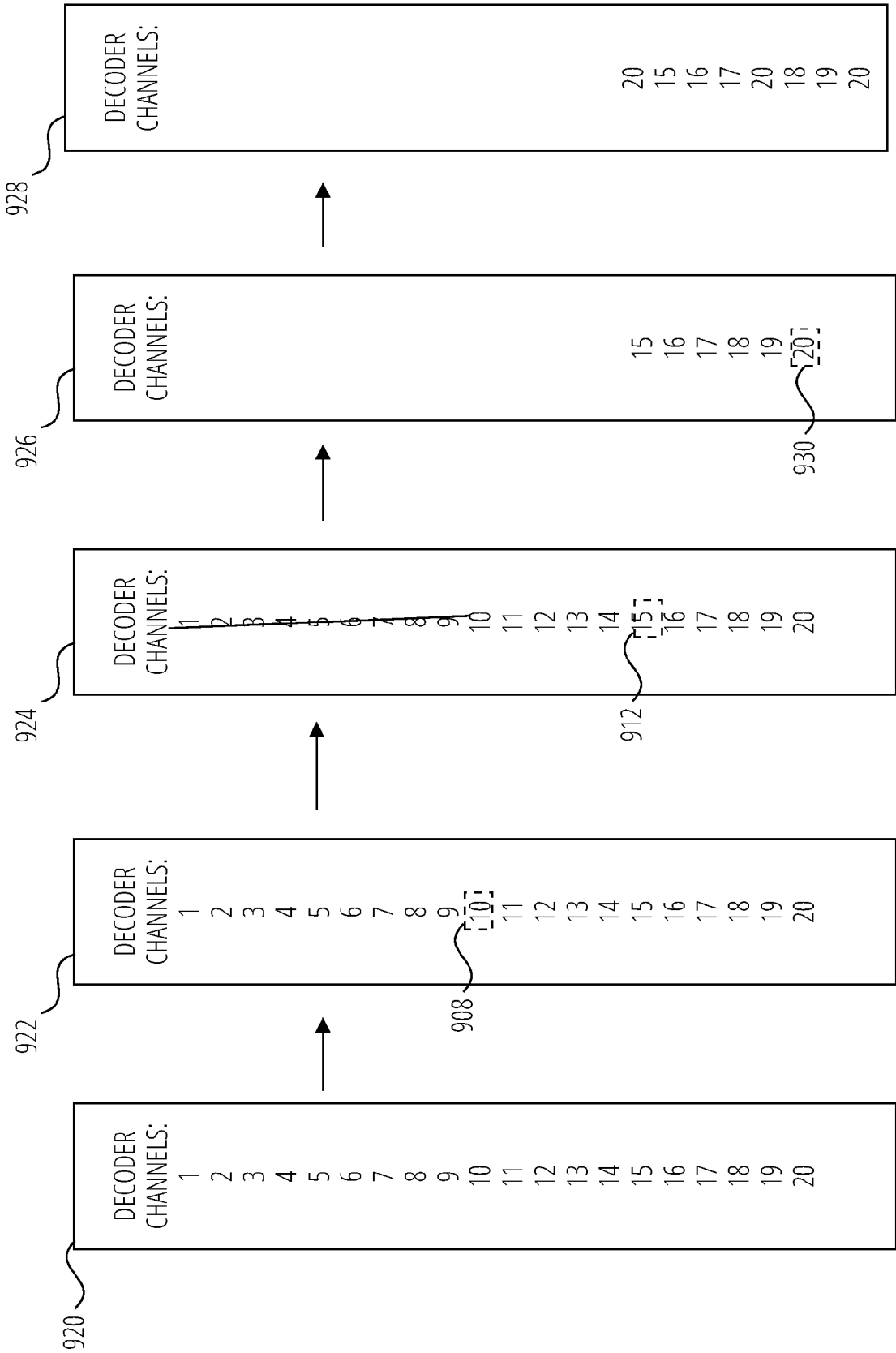
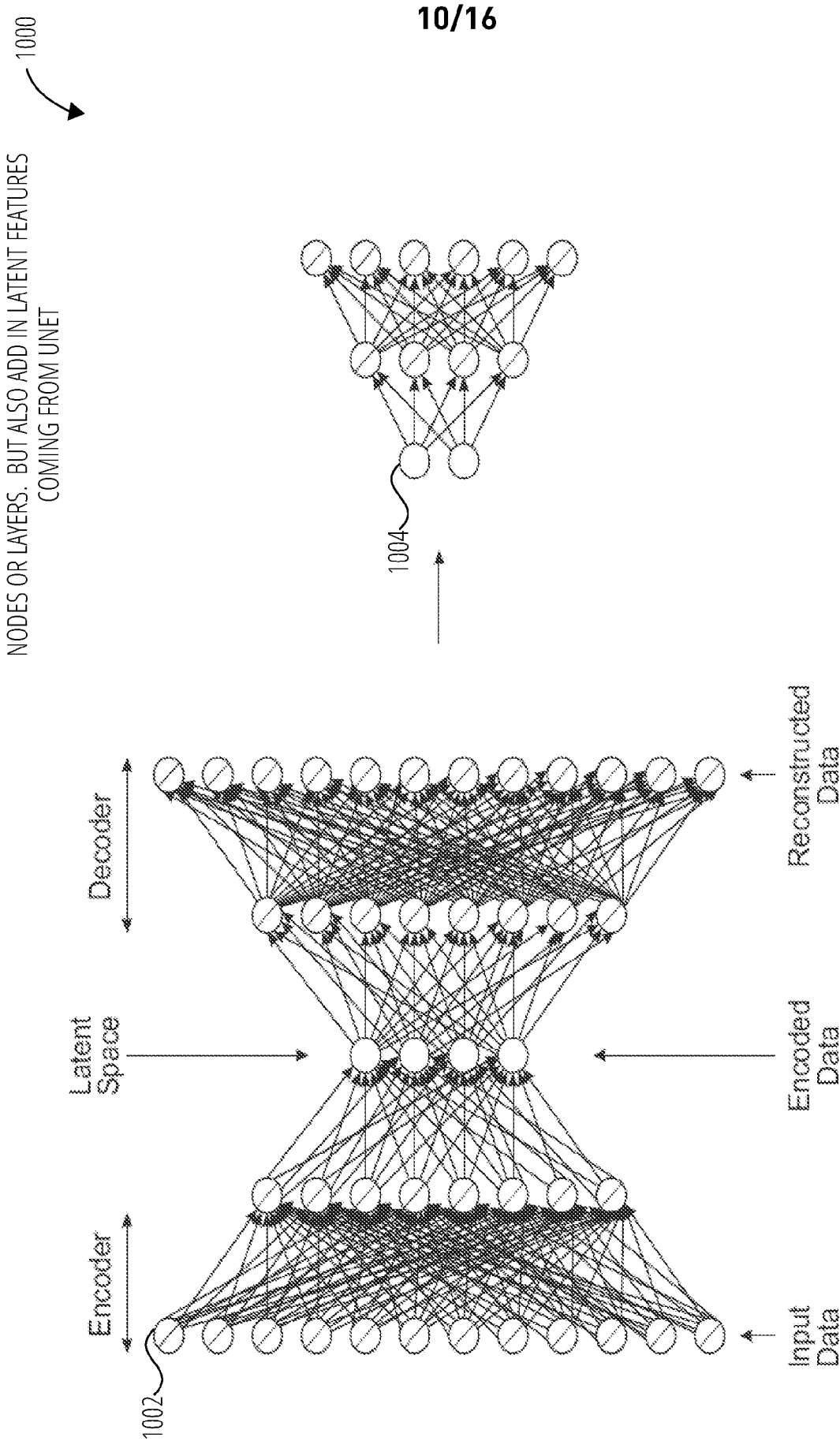


FIG. 9



10/16

FIG. 10

11/16

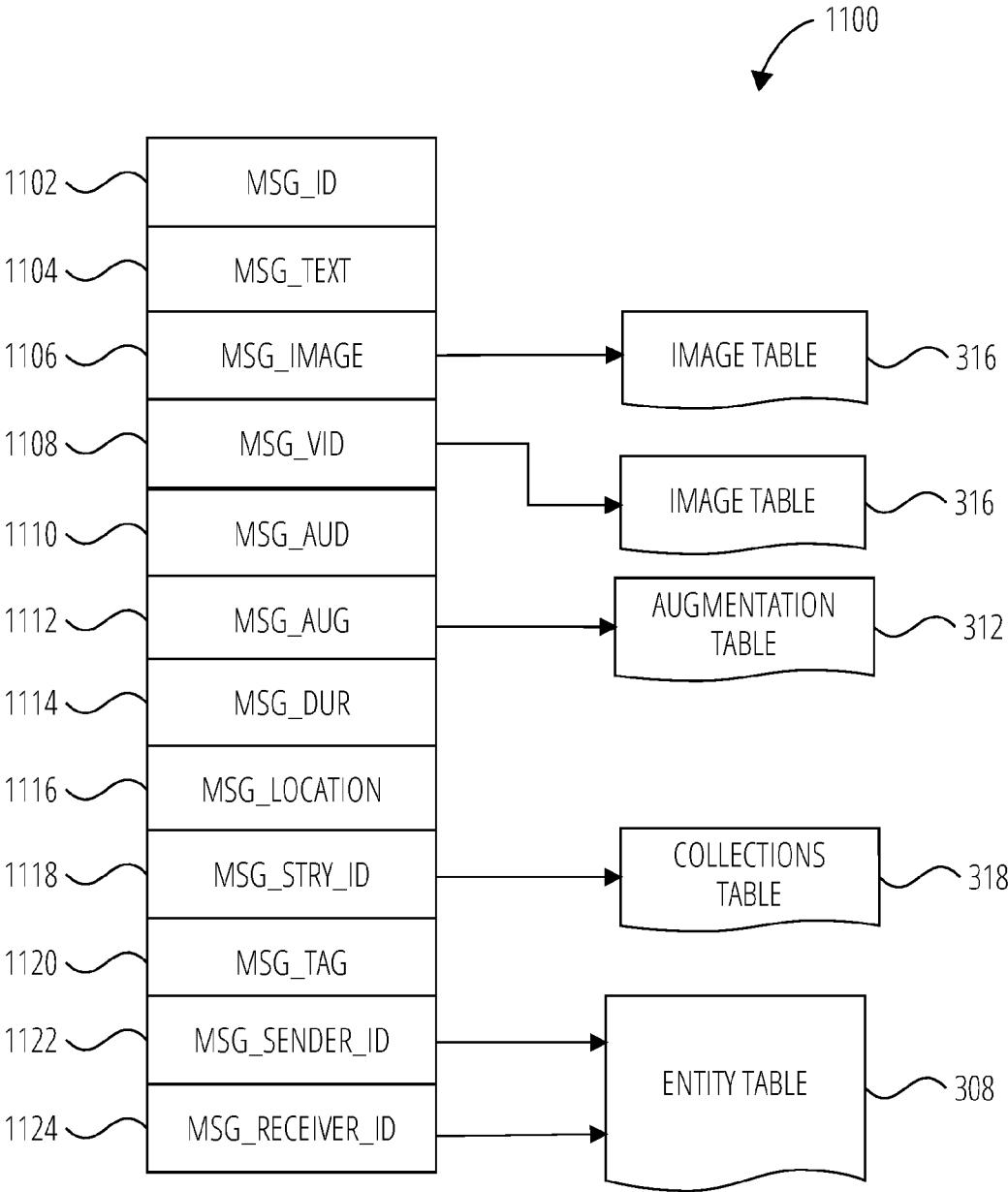
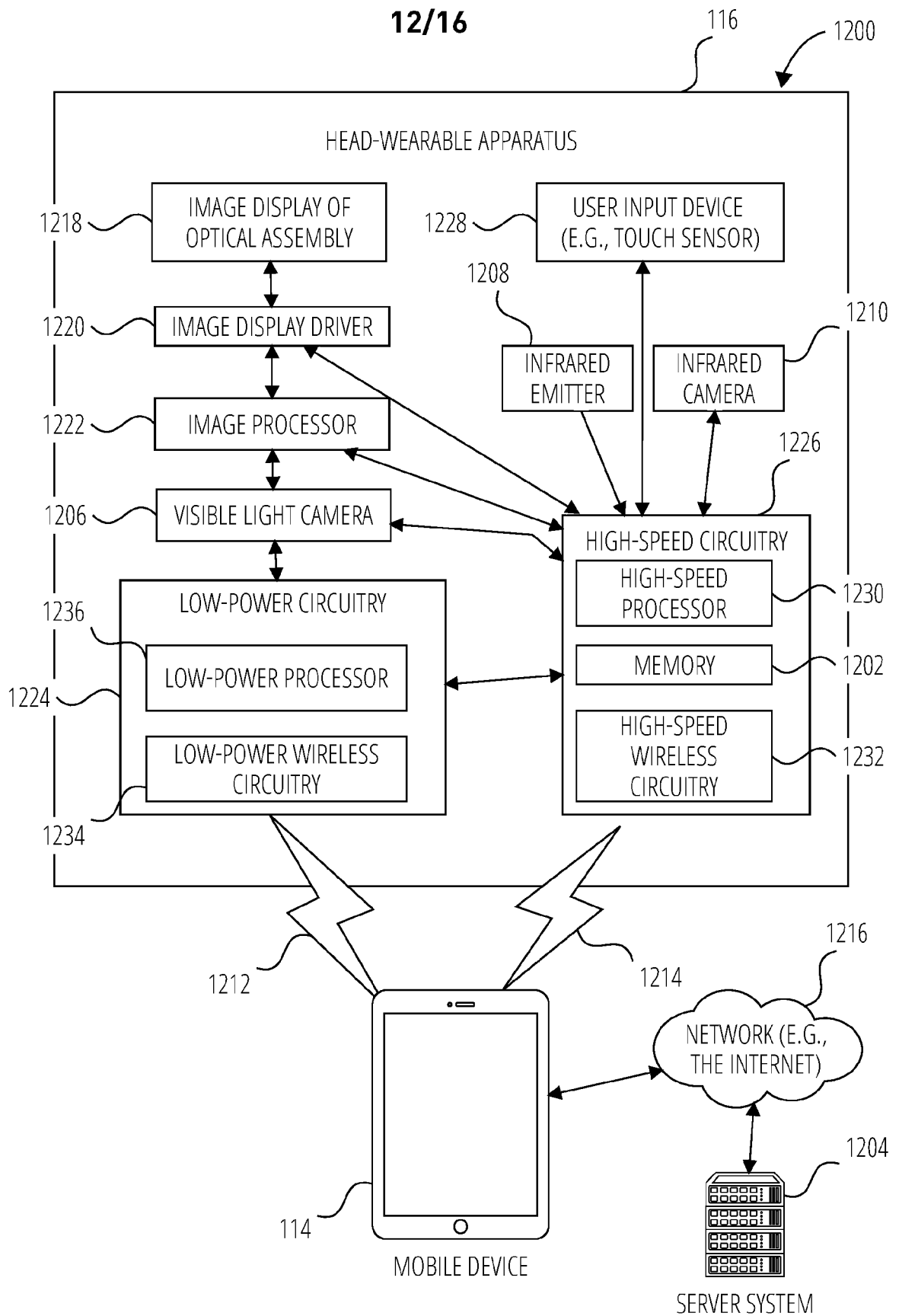
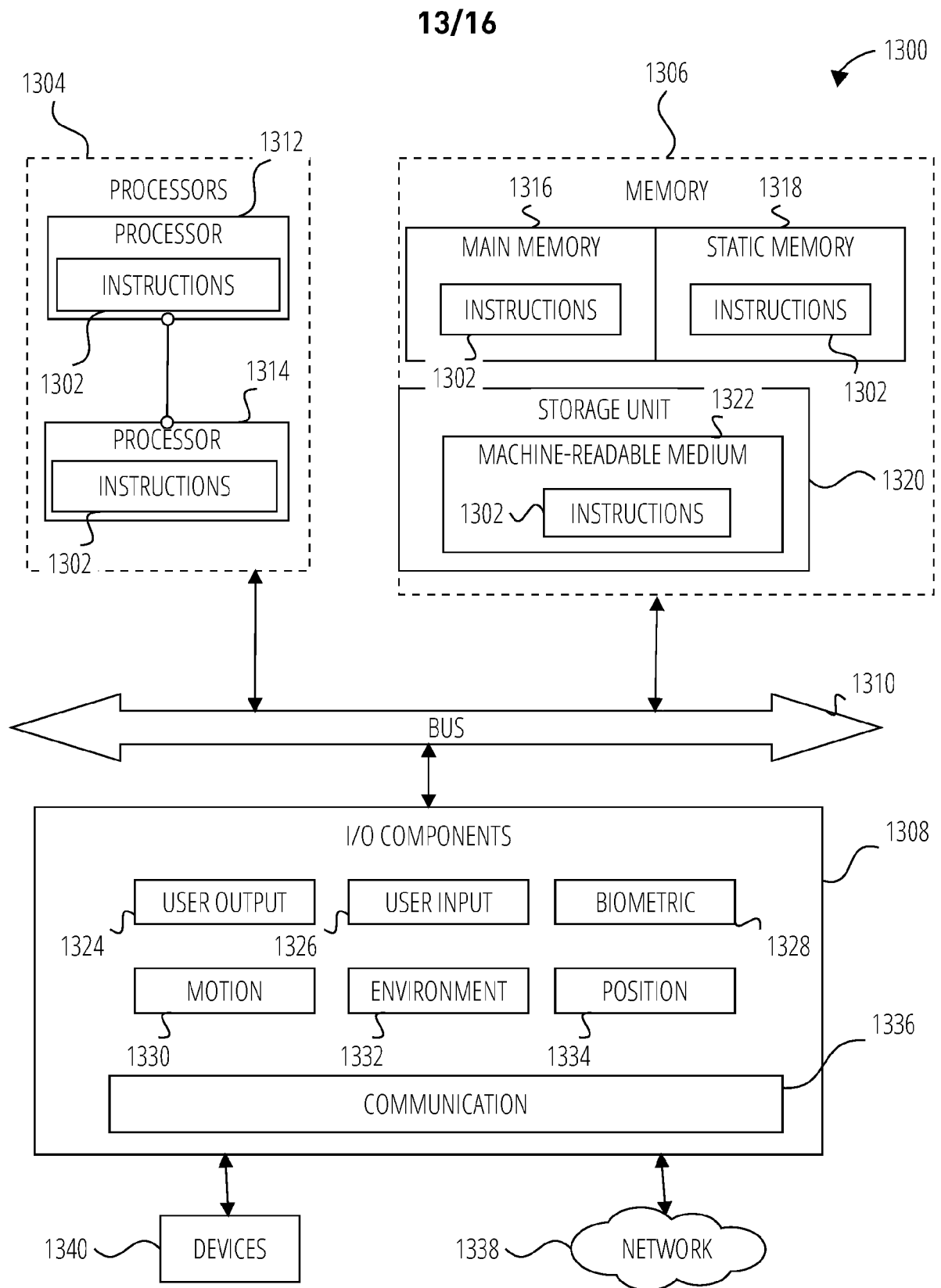
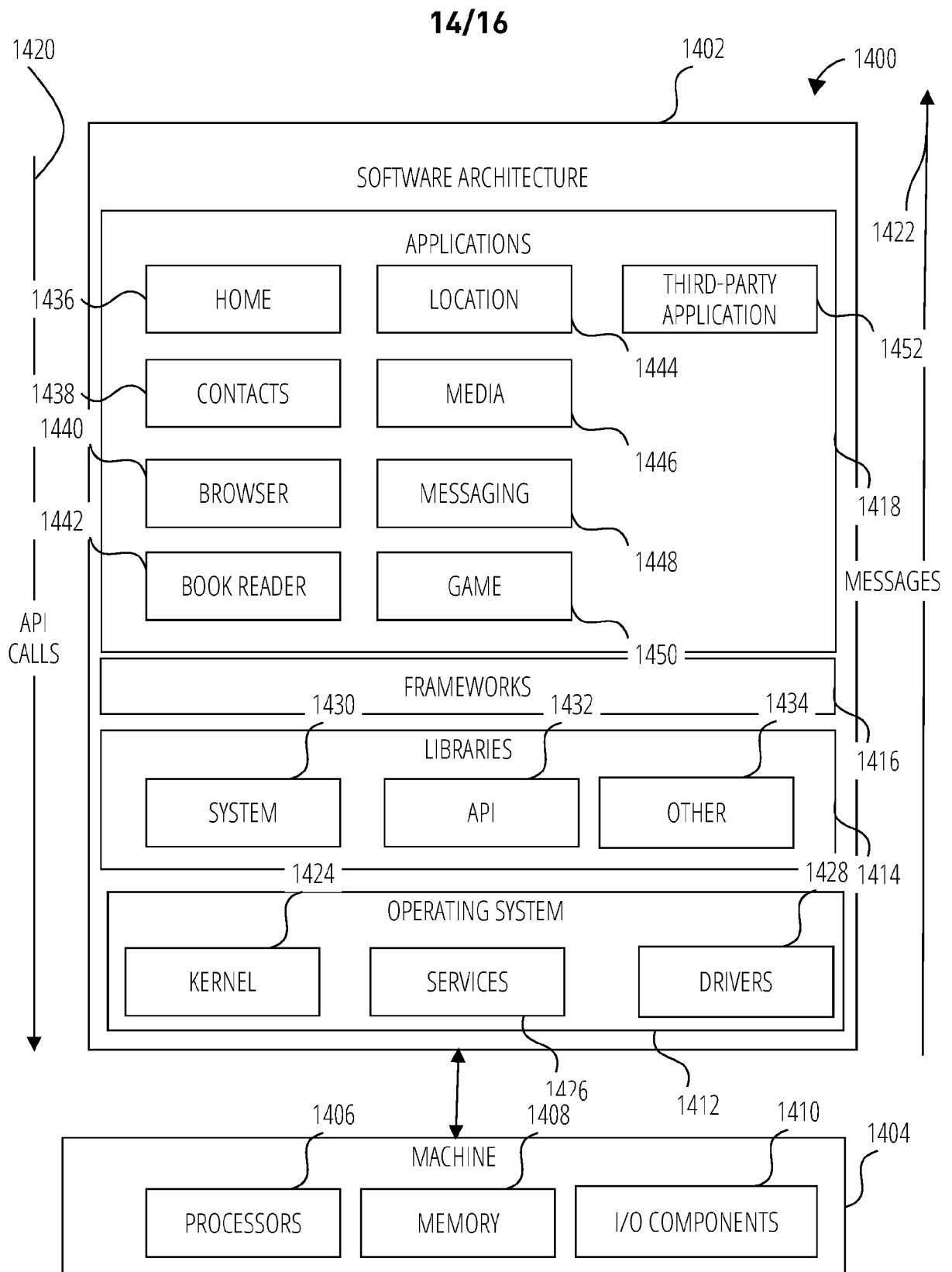
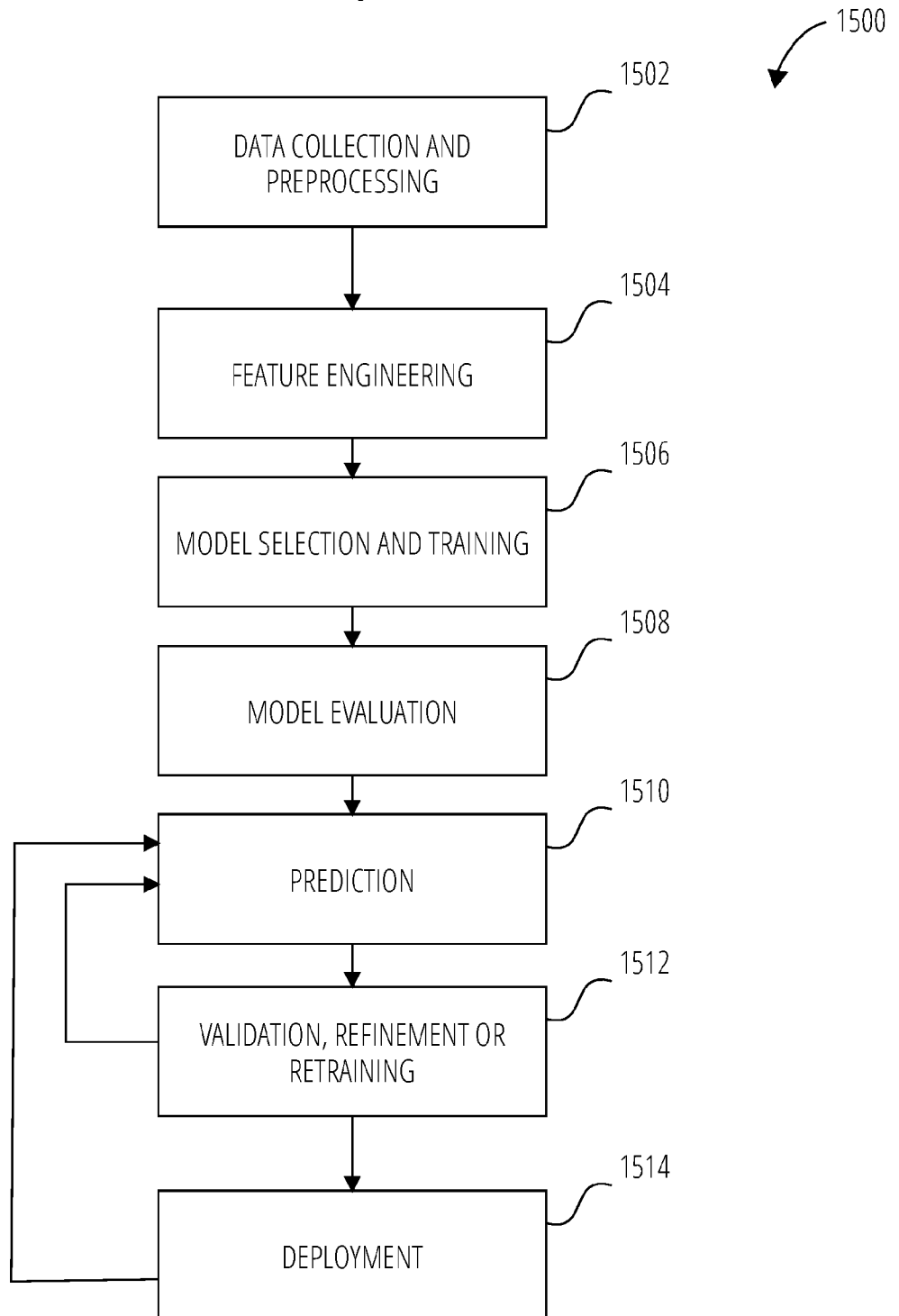


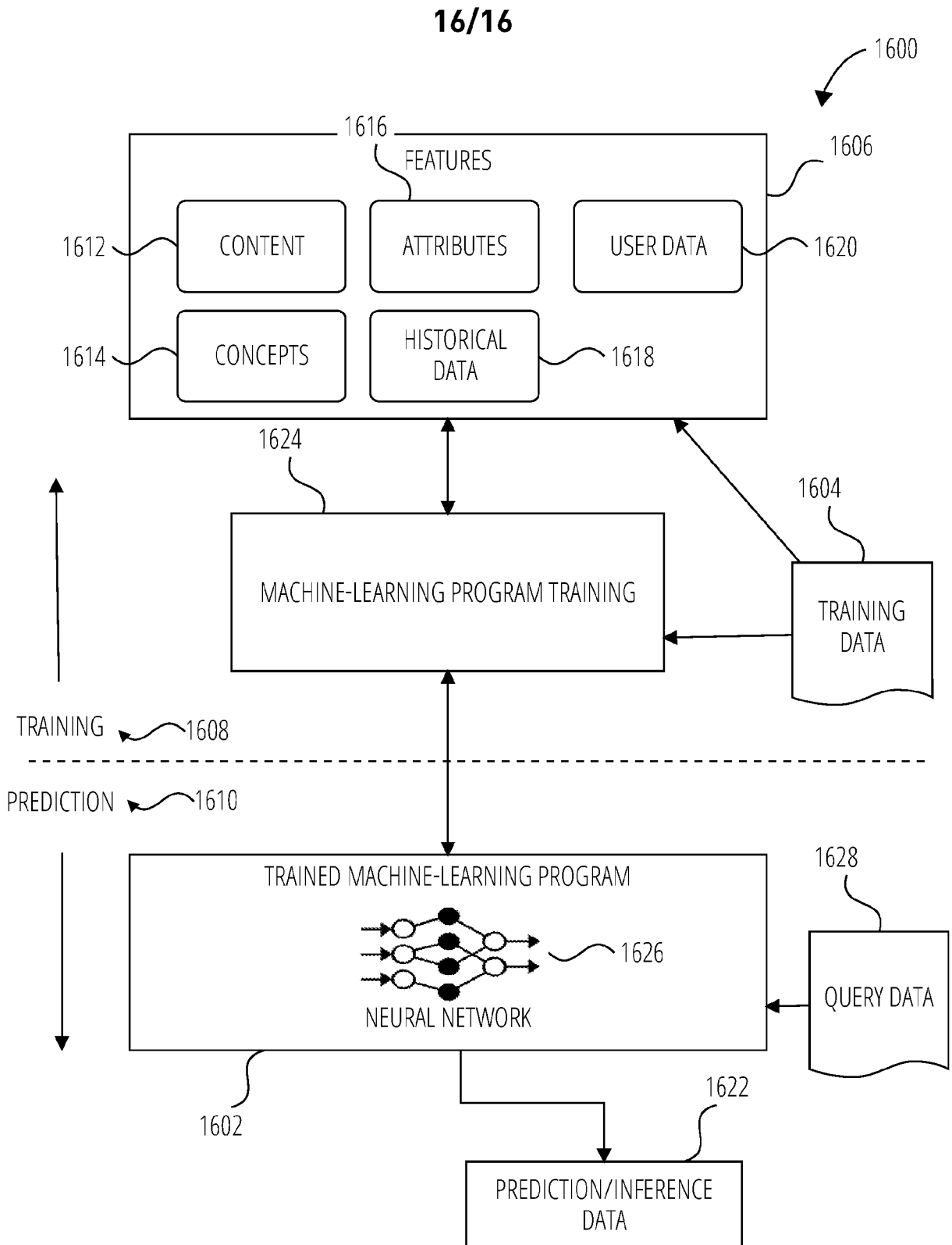
FIG. 11

**FIG. 12**

**FIG. 13**

**FIG. 14**

15/16**FIG. 15**

**FIG. 16**

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2024/030384

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06N3/0455 G06N3/082

ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N

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

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

EPO - Internal**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BO-KYEONG KIM ET AL: "On Architectural Compression of Text-to-Image Diffusion Models", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 25 May 2023 (2023-05-25), XP091518924, abstract; sections 3.2, 4.1, 4.2; figures 2, 3; table 3 ----- -/- -	1 - 20



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

22 August 2024

Date of mailing of the international search report

19/09/2024

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2

NL - 2280 HV Rijswijk

Tel. (+31-70) 340-2040,

Fax: (+31-70) 340-3016

Authorized officer

Tebbal Barracosa, B

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2024/030384

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	YANYU LI ET AL: "SnapFusion: Text-to-Image Diffusion Model on Mobile Devices within Two Seconds", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 1 June 2023 (2023-06-01), XP091526581, abstract; sections 3.1 Appendix B.2; figure 3	1-20
A	----- YUFAN ZHOU ET AL: "Shifted Diffusion for Text-to-image Generation", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 24 November 2022 (2022-11-24), XP091380788, abstract; sections 2, 3 algorithm 1; figure 2	1-20
A	----- ROBIN ROMBACH ET AL: "High-Resolution Image Synthesis with Latent Diffusion Models", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 20 December 2021 (2021-12-20), XP091118987, abstract; sections 3, 3.1, 3.2, 3.3, 4; figure 3 -----	1-20