

NETWORK-BASED VISUAL ANALYSIS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of priority of Singapore Patent Application No. 10201908371Q, filed on 11 September 2019, the content of which
5 being hereby incorporated by reference in its entirety for all purposes.

TECHNICAL FIELD

[0002] The present invention generally relates to network-based visual analysis,
10 and more particularly, to a method of visual data transmission for network-based visual analysis, a corresponding imaging device for visual data transmission for network-based visual analysis, a corresponding method of network-based visual analysis, a corresponding visual analysis device for network-based visual analysis, and a corresponding network-based visual analysis system.

15

BACKGROUND

[0003] With the advances of network infrastructure, recent years has witnessed the explosive growth of network-based (e.g., cloud-based) visual analysis applications, such as surveillance analysis, smart city, visual-based positioning, autopilot, and so on.
20 In cloud-based visual analysis, visual signals are acquired by the front end (which may interchangeably be referred to herein as front-end device(s), front-side device(s), edge-side device(s), edge device(s) or the like) and the analysis is completed in the server end (which may interchangeably be referred to as server(s), cloud end, cloud server(s), cloud-end server(s), cloud-side server(s) or the like). For example, front-end devices
25 may acquire information from users or the physical world, which may subsequently be transmitted over a wireless network to the server end (e.g., a data center) for further processing and analysis, such as shown in FIG 1. In particular, FIG. 1 depicts a schematic diagram of example network-based visual analysis applications. Images and videos may be acquired at the front end and the analysis may be performed at the server
30 end (e.g., cloud end). With deep learning models showing incomparable performance in computer vision (e.g., various computer vision tasks), visual analysis applications (e.g., cloud-based visual analysis) are increasingly relying on deep neural networks (DNNs), such as object detection, vehicle and person re-identification (ReID), vehicle

license plate recognition, face recognition, pedestrian detection, landmark retrieval, autopilot, and so on.

[0004] For data communication between the front end and the server end, the most conventional paradigm may be known as “compress-then-analyse”, such as shown in FIG. 2A. In particular, FIG. 2A illustrates visual signal transmission associated with the conventional “compress-then-analyse” approach. By transmitting the visual signal, a series of visual analysis tasks may be performed at the cloud end. As such, the computing load, including feature extraction and analysis, is imposed on the cloud side. Accordingly, the visual signal is captured and compressed in the front-end devices, then the coding bitstream is conveyed to the cloud-end server(s). Subsequently, the feature extraction and visual analysis tasks may be performed in the cloud-end server(s) according to the decoded visual signal. As the fundamental infrastructure of the paradigm, image/video compression has been well developed and matured. As the current generation video coding standard, High Efficiency Video Coding (HEVC) achieves half bit-rate reductions at equal perceptual visual quality level compared to the last generation H.264/MPEG-4 Advanced Video Coding (AVC). The next generation video coding standardization, Versatile Video Coding (VVC) is ongoing and has already achieved superior performance to HEVC.

[0005] Although supported by well developed standards and infrastructures, the “compress-then-analyse” paradigm is questionable when the system is scaled up. For example, in application scenarios such as Internet-of-Things (IoT) and video surveillance, thousands-of-thousands front-end cameras can simultaneously produce large amount of visual signals. The transmission bandwidth may be a bottleneck as signal level compression suffers from high transmission burden. Furthermore, the feature extraction of visual signals is computationally intensive, especially with deep neural networks, which makes it unaffordable to simultaneously analyse large scale visual data in the cloud-end servers. That is, the signal level visual compression imposes high transmission burden, while the computational load of the numerous deep learning models executed simultaneously for feature extraction also becomes a significant bottleneck at the cloud end.

[0006] FIG. 2B depicts another strategy “analyze-then-compress” for data communication between the front end and the server end. In particular, FIG. 2B illustrates ultimate feature (i.e., top-layer feature, such as a deep feature from a fully-connected layer of a deep neural network) transmission associated with the

conventional “analyze-then-compress” approach. Computing load can be distributed to front-end devices. With this strategy, both data acquisition and feature extraction occur in the front-end devices, and only the ultimately utilized features (i.e., top-layer features, which may interchangeably be referred to herein as ultimate features), instead of visual signals, are compressed and transmitted to the cloud end. In particular, FIG. 2B illustrates ultimate feature transmission associated with the conventional “analyze-then-compress” approach. With this approach, computing load can be distributed to front-end devices. However, only specific types of analysis can be performed at the server-end, depending on the deep models used at the front end. It provides a feasible solution for large scale cloud-based visual analysis systems, as the ultimate feature is compact and able to be utilized for analysis straightforwardly at the cloud end. Moreover, ultimate features may be extracted to reflect abstract semantic meaning, which largely eliminates visible information from the input signals.

[0007] As such, the risk of privacy disclosure may be controlled by conveying ultimate features instead of signal-level data communication. Such a paradigm is also supported by several feature coding standards for handcrafted ultimate features. In the context of image retrieval applications, Compact Descriptors for Visual Search (CDVS) was published by Moving Picture Experts Group (MPEG) in 2015. Built upon CDVS, Compact Descriptors standardization for Video Analysis (CDVA) was proposed by MPEG to deal with video retrieval applications.

[0008] For hand-crafted ultimate features, the standards from MPEG including MPEG-CDVS and MPEG-CDVA may specify the feature extraction and compression processes. For deep learning features, top-layer features (ultimate features, such as deep features from fully-connected layers of a deep neural network) of the deep learning models are transmitted to the cloud side, since the top-layer features of deep models are compact and can be straightforwardly utilized for analysis. For instance, in the face recognition task, the ultimate feature of a human face may be only with dimension of 4K in Facebook DeepFace, 128 in Google FaceNet, and 300 in Sense-Time DeepID3. In such scenarios, only the lightweight operations, such as feature comparison, are required to be performed at the cloud servers, while the heavy workloads of feature extraction are distributed to the front end. Moreover, transmitting ultimate features may also be favourable for privacy protection. In particular, instead of directly conveying the visual signal which may easily expose privacy, ultimate feature communication can largely avoid disclosing visible information

[0009] Although the data transmission strategy of conveying ultimate features may have a number of advantages, one obstacle that may hinder the practical implementation of ultimate feature communication is that ultimate features are usually task-specific which makes the transmitted features (ultimate features) hard to be applied to various analysis tasks. That is, one obstacle that may hinder the applications of deep learning feature compression is that deep learning models are normally designed and trained for specific tasks, and the ultimate features are extraordinary abstract and task-specific, making such compressed features (ultimate features) difficult to generalize. This may also impede the further standardization for deep feature coding, as standardized deep features may be required to be well generalized to ensure the interoperability in various application scenarios.

[0010] A need therefore exists to provide network-based visual analysis, such as a method of visual data transmission for network-based visual analysis and a corresponding method of network-based visual analysis, that seek to overcome, or at least ameliorates, one or more of the deficiencies in conventional network-based visual analysis, such as but not limited to, reducing the computational load at the server end in performing visual analysis without unduly or unsatisfactorily limiting usability or employability in the range of different types of visual analysis applications or tasks at the server end. It is against this background that the present invention has been developed.

SUMMARY

[0011] According to a first aspect of the present invention, there is provided a method of visual data transmission for network-based visual analysis, the method comprising:

obtaining, at an imaging device, sensor data relating to a scene;
extracting an intermediate deep feature from an intermediate layer of a deep learning model based on the sensor data;
producing encoded video data based on the intermediate deep feature; and
transmitting the encoded video data to a visual analysis device for performing visual analysis based on the encoded video data.

[0012] According to a second aspect of the present invention, there is provided a method of network-based visual analysis, the method comprising:

receiving, at a visual analysis device, encoded video data from an imaging device configured to obtain sensor data relating to a scene;

producing decoded video data based on the encoded video data;

5 producing an intermediate deep feature of a deep learning model based on the decoded video data; and

performing visual analysis based on the intermediate deep feature.

[0013] According to a third aspect of the present invention, there is provided an imaging device for visual data transmission for network-based visual analysis, the imaging device comprising:

10 a memory; and

at least one processor communicatively coupled to the memory and configured to perform the method of visual data transmission for network-based visual analysis according to the above-mentioned first aspect of the present invention.

[0014] According to a fourth aspect of the present invention, there is provided a visual analysis device for network-based visual analysis, the visual analysis device comprising:

a memory; and

15 at least one processor communicatively coupled to the memory and configured to perform the method of network-based visual analysis according to the above-mentioned second aspect of the present invention.

[0015] According to a fifth aspect of the present invention, there is provided a network-based visual analysis system, the network-based visual analysis system comprising:

20 one or more imaging devices, each imaging device being configured for visual data transmission for network-based visual analysis according to the above-mentioned third aspect of the present invention; and

a visual analysis device for network-based visual analysis configured according to the above-mentioned fourth aspect of the present invention, wherein the visual analysis device is configured to receive encoded video data from the one or more imaging devices, respectively.

30 **[0016]** According to a sixth aspect of the present invention, there is provided a computer program product, embodied in one or more non-transitory computer-readable storage mediums, comprising instructions executable by at least one processor to

perform a method of visual data transmission for network-based visual analysis according to the above-mentioned first aspect of the present invention.

[0017] According to a seventh aspect of the present invention, there is provided a computer program product, embodied in one or more non-transitory computer-readable storage mediums, comprising instructions executable by at least one processor to perform a method of network-based visual analysis according to the above-mentioned second aspect of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

10 [0018] Embodiments of the present invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example only, and in conjunction with the drawings, in which:

FIG. 1 depicts a schematic diagram of example network-based visual analysis applications;

15 FIG. 2A depicts visual signal transmission associated with a conventional “compress-then-analyse” approach;

FIG. 2B depicts ultimate feature (i.e., top-layer feature) transmission associated with a conventional “analyze-then-compress” approach;

20 FIG. 3 depicts a flow diagram of a method of visual data transmission for network-based visual analysis, according to various embodiments of the present invention;

FIG. 4 depicts a flow diagram of a method of network-based visual analysis, according to various embodiments of the present invention;

25 FIG. 5 depicts a schematic block diagram of an imaging device for visual data transmission for network-based visual analysis, according to various embodiments of the present invention;

FIG. 6 depicts a schematic block diagram of a visual analysis device for network-based visual analysis, according to various embodiments of the present invention;

30 FIG. 7 depicts an example portable computing device, which the imaging device as described with reference to FIG. 5 may be embodied in, by way of an example only;

FIG. 8 depicts a schematic block diagram of an exemplary computer system in which the visual analysis device as described with reference to FIG. 6 may be embodied in, by way of an example only;

FIG. 9 depicts a schematic block diagram of a network-based visual analysis system 900, according to various embodiments of the present invention;

FIG. 10 depicts a table (Table 1) comparing various attributes associated with three data transmission strategies or methods, namely, the conventional “compress-then-analyse” method (“Transmit Video Signal”), the “analyze-then-compress” method (“Transmit Ultimate Feature”) and the method of data transmission (“Transmit Intermediate Feature”) according to various example embodiments of the present invention;

FIG. 11 depicts a schematic drawing of a network-based (e.g., cloud-based) visual analysis system, according to various example embodiments of the present invention;

FIG. 12 depicts a table (Table 2) which summarizes the usability of the intermediate deep features, according to various example embodiments;

FIG. 13 depicts visualized feature maps of VGGNet, according to various example embodiments of the present invention;

FIGs. 14A and 14B depict schematic flow diagrams of network-based visual analysis, according to various example embodiments of the present invention;

FIGs. 15A to 15D depict plots illustrating distributions of feature maps of VGGNet-16 and ResNet-50, according to various example embodiments of the present invention;

FIG. 16 depicts an algorithm for a method of channel concatenation by distance, according to various example embodiments of the present invention;

FIG. 17A depicts a schematic drawing illustrating a method of channel concatenation by distance, according to various example embodiments of the present invention;

FIG. 17B depicts a schematic drawing illustrating a method of channel tiling, according to various example embodiments of the present invention;

FIG. 18 depicts an algorithm for a method of calculating similarity between two ranked sequences of documents, according to various example embodiments of the present invention

FIG. 19 depicts a table (Table 3) showing lossy feature compression results, according to various example embodiments of the present invention;

FIGs. 20A to 20E show plots comparing baseline, naïve channel concatenation, channel concatenation by distance and channel tiling, according to various example embodiments of the present invention;

FIG. 21 depicts a table (Table 4) showing the fidelity comparison of two pre-quantization methods (uniform and logarithmic) on different feature types and bit depth, according to various example embodiments of the present invention;

FIGs. 22A and 22B depict tables (Tables 5 and 6, respectively) listing lossy compression results on VGGNet-16 and ResNet-101, according to various example embodiments of the present invention; and

FIG. 23 depicts a schematic flow diagram of network-based visual analysis, according to various example embodiments of the present invention.

DETAILED DESCRIPTION

[0019] Various embodiments of the present invention relate to network-based visual analysis, and more particularly, to a method of visual data transmission for network-based visual analysis, a corresponding imaging device for visual data transmission for network-based visual analysis, a corresponding method of network-based visual analysis, a corresponding visual analysis device for network-based visual analysis, and a corresponding network-based visual analysis system. In various embodiments, a network-based visual analysis may refer to a visual analysis performed at least based on visual data transmitted over a network. In various embodiments, visual data may be any data comprising or formed based on sensor data relating to a scene obtained by an imaging device, such as still or video image data of the scene captured or sensed by an image sensor of a camera. In various embodiment, the network may be any wired or wireless communication network, such as but not limited to, Ethernet, cellular or mobile communication network (e.g., 3G, 4G, 5G or higher generation mobile communication network), Wi-Fi, wired or wireless sensor network, satellite communication network, wired or wireless personal or local area network, and so on. In various embodiments, the visual data may be encoded video data encoded based on any video encoding/decoding technology or technique, such as but not limited to, Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC) or Versatile Video Coding (VVC).

[0020] As discussed in the background, in relation to network-based visual analysis, for data communication between the front end and the server end,

conventional paradigms or approaches include the “compress-then-analyse” approach (e.g., as shown in FIG. 2A) or the “analyze-then-compress” approach (e.g., as shown in FIG. 2B). In relation to the “compress-then-analyse” approach, the signal level visual compression imposes high transmission burden, while the computational load of the numerous deep learning models executed simultaneously at the server end for feature extraction also becomes a significant bottleneck at the server end. In relation to the “analyze-then-compress” approach (as shown in FIG. 2B), the ultimate features (i.e., top-layer features, such as deep features from fully-connected layers of a deep neural network, which are in the form of one-dimensional (1D) arrays (which may also be referred to as 1D feature vectors)) are extraordinary abstract and task-specific, making such compressed features difficult to generalize, thus hindering the practical implementation of the transmitted ultimate features in various visual analysis applications or tasks.

[0021] Accordingly, various embodiments of the present invention provide network-based visual analysis, such as a method of visual data transmission for network-based visual analysis and a corresponding method of network-based visual analysis, that seek to overcome, or at least ameliorates, one or more of the deficiencies in conventional network-based visual analysis, such as but not limited to, reducing the computational load at the server end in performing visual analysis without unduly or unsatisfactorily limiting (e.g., without or minimizes limiting) usability or employability in the range of different types of visual analysis applications or tasks at the server end.

[0022] FIG. 3 depicts a flow diagram of a method 300 of visual data transmission for network-based visual analysis according to various embodiments of the present invention. The method 300 comprising: obtaining (at 302), at an imaging device, sensor data relating to a scene; extracting (at 304) an intermediate deep feature from an intermediate layer of a deep learning model based on the sensor data; producing (at 306) encoded video data based on the intermediate deep feature; and transmitting (at 308) the encoded video data to a visual analysis device for performing visual analysis based on the encoded video data.

[0023] In various embodiments, in relation to 302, the sensor data relating to a scene obtained by the imaging device may be still or video image data of the scene captured or sensed by an image sensor of the imaging device. In various embodiments, the imaging device may be any device (which may also be embodied as a system or an apparatus) having an image capturing component or unit (e.g., an image sensor),

communication functionality or capability (e.g., wired or wireless communication interface), a memory and at least one processor communicatively coupled to the memory, such as but not limited to, a smartphone, a wearable device (e.g., a smart watch, a head-mounted display (HDM) device, and so on), and a camera (e.g., a portable camera, a surveillance camera, a vehicle or dashboard camera, and so on).

[0024] In various embodiments, in relation to 304, the deep learning model may be a deep neural network, such as a convolutional neural network (CNN) comprising an input layer, convolutional layer(s), fully-connected layer(s) and an output layer. An intermediate layer of a deep learning model can be understood by a person skilled in the art. For example, an intermediate layer of a CNN may correspond to one of the convolutional layer(s). Accordingly, an intermediate feature is a feature obtained (extracted) from an intermediate layer of a deep learning model, which is in the form of multi-dimensional arrays (i.e., two or more dimensions). In various embodiments, the intermediate feature comprises a plurality of feature maps, each feature map in the form of a two-dimensional (2D) array. For example, activations (e.g., by an activation function, such as a rectified linear unit (ReLU)) from an intermediate layer may be considered as or constitute a plurality of feature maps. The sensor data may be input to an input layer of the deep neural network.

[0025] In various embodiments, in relation to 306, the encoded video data may be encoded by any video encoding/decoding technique or technology, such as but not limited to, Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC) or Versatile Video Coding (VVC).

[0026] In various embodiments, in relation to 308, the encoded video data may be transmitting via any wired or wireless communication network, such as but not limited to, Ethernet, cellular or mobile communication network (e.g., 3G, 4G, 5G or higher generation mobile communication network), Wi-Fi, wired or wireless sensor network, satellite communication network, wired or wireless personal or local area network, and so on.

[0027] Accordingly, the method 300 of visual data transmission for network-based visual analysis advantageously reduces the computational load at the visual analysis device (e.g., the server end) in performing visual analysis without unduly or unsatisfactorily limiting (e.g., without or minimizes limiting) usability or employability in the range of different types of visual analysis applications or tasks at the visual analysis device. In particular, encoded video data based on an intermediate deep feature

from an intermediate layer of a deep learning model based on sensor data is advantageously transmitted to the visual analysis device for performing visual analysis based on the encoded video data. These advantages or technical effects will become more apparent to a person skilled in the art as the network-based visual analysis is described in more detail according to various embodiments or example embodiments of the present invention.

5 [0028] In various embodiments, the encoded video data is produced based on a video codec. In various embodiments, the video codec may be based on any video encoding/decoding technique or technology as desired or as appropriate, such as but not limited to, Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC) or Versatile Video Coding (VVC).

10 [0029] In various embodiments, the intermediate deep feature comprises a plurality of feature maps. In this regard, the method 300 further comprises producing video format data based on the plurality of feature maps, and the above-mentioned producing (at 306) encoded video data comprises encoding the video format data using the video codec to produce the encoded video data. In various embodiments, the video format data may be any data configured to fit or suitable for an input of the video codec for the video codec to encode the video format data into the encoded video data, such as a video sequence format data (e.g., YUV400 format data).

15 [0030] In various embodiments, the above-mentioned producing video format data comprises repacking the plurality of feature maps based on a repacking technique to produce the video format data. In various embodiments, the repacking technique may be configured to group or organize (or regroup or reorganize) the plurality of feature maps into an ordered plurality of feature maps, resulting in a video format data. For example, the ordered plurality of feature maps may have the same or a different order as before. For example, the repacking technique may be configured to improve the coding efficiency of the video codec with respect to the video format data inputted thereto.

20 [0031] In various embodiments, the repacking technique is based on channel concatenation or channel tiling. In various embodiments, the channel concatenation may be a naïve channel concatenation technique or a channel concatenation by distance technique. The naïve channel concatenation technique or the channel concatenation by distance technique will be described in further detail later below according to various example embodiments of the present invention.

[0032] In various embodiments, the repacking technique is based on the above-mentioned channel concatenation, and more particular, the channel concatenation by distance technique. In this regard, the above-mentioned channel concatenation comprising determining a plurality of inter-channel distances associated with the plurality of feature maps, each inter-channel distance being associated with a pair of feature maps of the plurality of feature maps, and the above-mentioned repacking the plurality of feature maps comprising forming a plurality of repacked feature maps by ordering the plurality of feature maps based on the plurality of inter-channel distances determined to produce the video format data comprising the plurality of repacked feature maps. In various embodiments, the plurality of repacked feature maps may simply refer to the resultant plurality of feature maps that has been repacked by the repacking technique. In various example embodiments, an inter-channel distance may be determined for each unique pair of feature maps of the plurality of feature maps.

[0033] In various embodiments, the repacking technique is based on the above-mentioned channel tiling, and the above-mentioned channel tiling comprising forming one or more repacked feature maps based on the plurality of feature maps to produce the video format data comprising the one or more repacked feature maps, each repacked feature map being an enlarged feature map. In various embodiments, the one or more repacked feature maps may simply refer to the resultant one or more feature maps that has been repacked by the repacking technique. In various embodiments, the enlarged feature map may be formed by tiling or joining two or more of the plurality of the plurality of feature maps in a planar manner to form an enlarged 2D array.

[0034] In various embodiments, the method 300 further comprises quantizing (which may also be interchangeably referred to as prequantizing) the plurality of feature maps to obtain a plurality of quantized feature maps (which may also be interchangeably referred to as a plurality of prequantized feature maps), respectively. In this regard, the video format data is produced based on the plurality of quantized feature maps. In various embodiments, the quantization may be performed to modify a numerical type of the plurality of feature maps from a floating point format to an integer format and/or to reduce the data volume of the plurality of feature maps.

[0035] In various embodiments, the method 300 further comprises: determining whether the plurality of feature maps are in a floating point format or in an integer format; and quantizing the plurality of feature maps to obtain a plurality of quantized feature maps, respectively, if the plurality of feature maps are determined to be in the

floating point format. In this regard, the video format data is produced based on the plurality of feature maps, without said the above-mentioned quantizing the plurality of feature maps, if the plurality of feature maps are determined to be in the integer format, or based on the plurality of quantized feature maps if the plurality of feature maps are determined to be in the floating point format. That is, the plurality of feature maps may be modified or converted into an integer format if they are in a floating point format, otherwise (i.e., if the plurality of feature maps are already in an integer format), the above-mentioned quantizing the plurality of feature maps may be skipped. In various embodiments, the numerical type (e.g., floating point format or integer format) of the plurality of feature maps may be determined based on a numerical type information (e.g., a flag or an identifier) associated with the plurality of feature maps.

[0036] In various embodiments, the plurality of feature maps are quantized based on a uniform quantization technique, a logarithmic quantization technique or a learning-based adaptive quantization technique. The uniform quantization technique, the logarithmic quantization technique or the learning-based adaptive quantization technique will be described in further detail later below according to various example embodiments of the present invention.

[0037] FIG. 4 depicts a flow diagram of a method 400 of network-based visual analysis. The method 400 comprises: receiving (at 402), at a visual analysis device, encoded video data from an imaging device configured to obtain sensor data relating to a scene; producing (at 404) decoded video data based on the encoded video data; producing (at 406) an intermediate deep feature of a deep learning model based on the decoded video data; and performing (at 408) visual analysis based on the intermediate deep feature.

[0038] In various embodiments, the method 400 of the network-based visual analysis corresponds to the method 300 of visual data transmission for network-based visual analysis described hereinbefore according to various embodiments of the present invention. Therefore, various functions or operations of the method 400 correspond to (e.g., are inverse of) various functions or operations of the method 300 described hereinbefore according to various embodiments. In other words, various embodiments described herein in context of the method 300 are correspondingly valid (e.g., being an inverse of) for the corresponding method 400, and vice versa. In particular, the method 300 of visual data transmission for network-based visual analysis and the method 400 of the network-based visual analysis may correspond to an encoding process or phase

and a decoding process or phase of the network-based visual analysis, and thus, it will be appreciated that various That is, in general, various functions or operations of the method 400 are inverse of various functions or operations of the method 300 described hereinbefore according to various embodiments.

5 **[0039]** Accordingly, the method 400 of visual data method 400 of the network-based visual analysis advantageously reduces the computational load at the visual analysis device (e.g., the server end) in performing visual analysis without unduly or unsatisfactorily limiting (e.g., without or minimizes limiting) usability or employability in the range of different types of visual analysis applications or tasks at the visual
10 analysis device. In particular, encoded video data based on an intermediate deep feature from an intermediate layer of a deep learning model based on sensor data is advantageously received by the visual analysis device for performing visual analysis based on the encoded video data. These advantages or technical effects will become more apparent to a person skilled in the art as the network-based visual analysis is
15 described in more detail according to various embodiments or example embodiments of the present invention.

[0040] In various embodiments, the above-mentioned producing (at 404) decoded video data comprises decoding the encoded video data using a video codec to produce the decoded video data comprising video format data. In various embodiments,
20 similarly, the video codec may be based on any video encoding/decoding technique or technology as desired or as appropriate, such as but not limited to, Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC) or Versatile Video Coding (VVC). In various embodiments, the video format data may correspond to (e.g., the same as) the video format data produced by the method 300.

25 **[0041]** In various embodiments, the intermediate deep feature comprises a plurality of feature maps.

[0042] In various embodiments, the producing (at 406) an intermediate deep feature comprises de-repackaging the video format data based on a de-repackaging technique to produce a plurality of de-repacked feature maps, and the intermediate deep feature is
30 produced based on the plurality of de-repacked feature maps. In various embodiments, the de-repackaging technique may be an inverse of the repackaging technique in the method 300, which restores the video format data (e.g., corresponding to the video format data comprising the ordered plurality of feature maps in the method 300) back to the original

order or configuration of the plurality of feature maps (which is referred to in the method 400 as the plurality of de-repacked feature maps).

[0043] In various embodiments, the de-repacking technique is based on channel de-concatenation or channel de-tiling. In various embodiments, the channel de-concatenation technique may be an inverse of the channel concatenation technique in the method 300, and the channel de-tiling technique may be an inverse of the channel tiling technique in the method 300.

[0044] In various embodiments, the video format data comprises a plurality of repacked feature maps (e.g., corresponding to the video format data comprising the ordered plurality of feature maps produced by the channel concatenation in the method 300). In this regard, the de-repacking technique is based on the above-mentioned channel de-concatenation, the above-mentioned channel de-concatenation comprising sorting the plurality of repacked feature maps based on repacking supplemental information to produce the plurality of de-repacked feature maps. In various embodiments, sorting the plurality of repacked feature maps may be restoring the plurality of repacked feature maps back to the original order of the plurality of feature maps based on the repacking supplemental information.

[0045] In various embodiments, the video format data comprises one or more repacked feature maps (e.g., corresponding to the video format data comprising one or more enlarged feature maps produced by the channel tiling in the method 300). In this regard, the de-repacking technique is based on the above-mentioned channel de-tiling, the above-mentioned channel de-tiling comprising forming the plurality of de-repacked feature maps based on the one or more repacked feature maps, each de-repacked feature map being a diminished feature map. In various embodiments, forming the plurality of repacked feature maps may be restoring the one or more repacked feature maps back to the original configuration of the plurality of feature maps based on repacking supplemental information.

[0046] In various embodiments, the method 400 further comprises de-quantizing (which may also be interchangeably referred to as de-prequantizing) the plurality of de-repacked feature maps to obtain a plurality of de-quantized feature maps (which may also be interchangeably referred to as a plurality of de-prequantized feature maps), respectively. In this regard, the intermediate deep feature is produced based on the plurality of de-quantized feature maps. In various embodiments, the de-quantizing technique may be an inverse of the quantizing technique in the method 300. In various

embodiments, the de-quantizing technique may be performed to modify the numerical type of the plurality of de-repacked feature maps from the integer format back to the floating point format.

[0047] In various embodiments, the method 400 further comprises: determining
5 whether the plurality of de-repacked feature maps are based on a plurality of original feature maps in a floating point format or in an integer format; and de-quantizing the plurality of de-repacked feature maps to obtain a plurality of de-quantized feature maps, respectively, if the plurality of de-repacked feature maps are determined to be based on the plurality of original feature maps in the floating point format. In this regard, the
10 intermediate deep feature is produced based on the plurality of de-repacked feature maps, without the above-mentioned de-quantizing the plurality of de-repacked feature maps, if the plurality of de-repacked feature maps are determined to be based on the plurality of original feature maps in the integer format or based on the plurality of de-quantized feature maps if the plurality of de-repacked feature maps are determined to be based on the plurality of original feature maps in the floating point format. That is,
15 the plurality of de-repacked feature maps may be modified or restored back into a floating point format if the plurality of de-repacked feature maps are based on a plurality of original feature maps (e.g., corresponding to the plurality of feature map in the intermediate deep feature extracted in the method 300) in a floating point format, otherwise (i.e., if the plurality of de-repacked feature maps are based on a plurality of
20 original feature maps are based on a plurality of original feature maps in an integer format (i.e., already in an integer format), the above-mentioned de-quantizing the plurality of de-repacked feature maps may be skipped. In various embodiments, similarly, the numerical type (e.g., floating point format or integer format) of the plurality of original feature maps may be determined based on a numerical type
25 information (e.g., a flag or an identifier) associated with the plurality of feature maps and transmitted to the visual analysis device.

[0048] In various embodiments, the plurality of de-repacked feature maps are de-quantized based on a uniform de-quantization technique, a logarithmic de-quantization
30 technique or a learning-based adaptive de-quantization technique. The uniform de-quantization technique, the logarithmic de-quantization technique or the learning-based adaptive de-quantization technique will be described in further detail later below according to various example embodiments of the present invention.

[0049] FIG. 5 depicts a schematic block diagram of an imaging device 500 for visual data transmission for network-based visual analysis according to various embodiments of the present invention, corresponding to the method 300 of visual data transmission for network-based visual analysis as described hereinbefore according to various embodiments of the present invention. The imaging device 500 comprises a memory 502, and at least one processor 504 communicatively coupled to the memory 502 and configured to perform the method 300 of visual data transmission for network-based visual analysis as described hereinbefore according to various embodiments of the present invention. In various embodiments, the at least one processor 504 is configured to: obtain sensor data relating to a scene; extract an intermediate deep feature from an intermediate layer of a deep learning model based on the sensor data; producing encoded video data based on the intermediate deep feature; and transmitting the encoded video data to a visual analysis device for performing visual analysis based on the encoded video data.

[0050] It will be appreciated by a person skilled in the art that the at least one processor 504 may be configured to perform the required functions or operations through set(s) of instructions (e.g., software modules) executable by the at least one processor 504 to perform the required functions or operations. Accordingly, as shown in FIG. 5, the imaging device 500 may comprise a sensor data obtaining module (or a sensor data obtaining circuit) 506 configured to obtain sensor data relating to a scene; an intermediate deep feature extracting module (or an intermediate deep feature extracting circuit) 508 configured to extract an intermediate deep feature from an intermediate layer of a deep learning model based on the sensor data; a video data encoding module 510 configured to produce encoded video data based on the intermediate deep feature; and a encoded video data transmitting module 512 configured to transmit the encoded video data to a visual analysis device (e.g., the visual analysis device 600) for performing visual analysis based on the encoded video data.

[0051] It will be appreciated by a person skilled in the art that the above-mentioned modules are not necessarily separate modules, and one or more modules may be realized by or implemented as one functional module (e.g., a circuit or a software program) as desired or as appropriate without deviating from the scope of the present invention. For example, the sensor data obtaining module 506, the intermediate deep feature extracting module 508, the video data encoding module 510 and the encoded video data transmitting module 512 may be realized (e.g., compiled together) as one

executable software program (e.g., software application or simply referred to as an “app”), which for example may be stored in the memory 502 and executable by the at least one processor 504 to perform the functions/operations as described herein according to various embodiments. In various embodiments, the encoded video data
5 transmitting module 512 may be configured to transmit the encoded video data to a visual analysis device via a wired or wireless signal transmitter or a transceiver of the imaging device 500.

[0052] In various embodiments, the imaging device 500 corresponds to the method 300 of visual data transmission for network-based visual analysis as described
10 hereinbefore with reference to FIG. 3, therefore, various functions or operations configured to be performed by the least one processor 504 may correspond to various steps of the method 300 described hereinbefore according to various embodiments, and thus need not be repeated with respect to the imaging device 500 for clarity and conciseness. In other words, various embodiments described herein in context of the
15 methods are analogously valid for the respective devices/systems (e.g., the imaging device 500), and vice versa.

[0053] For example, in various embodiments, the memory 502 may have stored therein the sensor data obtaining module 506, the intermediate deep feature extracting module 508, the video data encoding module 510 and/or the encoded video data
20 transmitting module 512, which respectively correspond to various steps of the method 300 as described hereinbefore according to various embodiments, which are executable by the at least one processor 504 to perform the corresponding functions/operations as described herein.

[0054] FIG. 6 depicts a schematic block diagram of a visual analysis device 600 for
25 network-based visual analysis according to various embodiments of the present invention, corresponding to the method 400 of network-based visual analysis as described hereinbefore according to various embodiments of the present invention. The visual analysis device 600 comprising: a memory 602; and at least one processor 604 communicatively coupled to the memory 602 and configured to perform the method
30 400 of network-based visual analysis as described hereinbefore according to various embodiments of the present invention. In various embodiments, the at least one processor 604 is configured to: receiving encoded video data from an imaging device configured to obtain sensor data relating to a scene; producing decoded video data based on the encoded video data; producing an intermediate deep feature of a deep learning

model based on the decoded video data; and performing visual analysis based on the intermediate deep feature.

[0055] Similarly, it will be appreciated by a person skilled in the art that the at least one processor 604 may be configured to perform the required functions or operations through set(s) of instructions (e.g., software modules) executable by the at least one processor 604 to perform the required functions or operations. Accordingly, as shown in FIG. 6, the visual analysis device 600 may comprise: an encoded video data receiving module (or an encoded video data receiving circuit) 606 configured to receive encoded video data from an imaging device (e.g., the imaging device 500) configured to obtain sensor data relating to a scene; a video data decoding module (or a video data decoding circuit) 608 configured to produce decoded video data based on the encoded video data; an intermediate deep feature producing module (or an intermediate deep feature producing circuit) 610 configured to produce an intermediate deep feature of a deep learning model based on the decoded video data; and a visual analysis performing module (or a visual analysis performing circuit) 612 configured to perform visual analysis based on the intermediate deep feature.

[0056] Similarly, it will be appreciated by a person skilled in the art that the above-mentioned modules are not necessarily separate modules, and one or more modules may be realized by or implemented as one functional module (e.g., a circuit or a software program) as desired or as appropriate without deviating from the scope of the present invention. For example, the encoded video data receiving module 606, the video data decoding module 608, the intermediate deep feature producing module 610 and the visual analysis performing module 612 may be realized (e.g., compiled together) as one executable software program (e.g., software application or simply referred to as an “app”), which for example may be stored in the memory 602 and executable by the at least one processor 604 to perform the functions/operations as described herein according to various embodiments. In various embodiments, the encoded video data receiving module 606 may be configured to receive the encoded video data from an imaging device via a wired or wireless signal receiver or a transceiver of the visual analysis device 600.

[0057] In various embodiments, the visual analysis device 600 corresponds to the method 400 of network-based visual analysis as described hereinbefore with reference to FIG. 4, therefore, various functions or operations configured to be performed by the least one processor 604 may correspond to various steps of the method 600 described

hereinbefore according to various embodiments, and thus need not be repeated with respect to the visual analysis device 600 for clarity and conciseness. In other words, various embodiments described herein in context of the methods are analogously valid for the respective devices/systems (e.g., the visual analysis device 600), and vice versa.

5 **[0058]** For example, in various embodiments, the memory 502 may have stored therein the encoded video data receiving module 606, the video data decoding module 608, the intermediate deep feature producing module 610 and/or the visual analysis performing module 612, which respectively correspond to various steps of the method 400 as described hereinbefore according to various embodiments, which are executable
10 by the at least one processor 604 to perform the corresponding functions/operations as described herein.

[0059] A computing system, a controller, a microcontroller or any other system providing a processing capability may be provided according to various embodiments in the present disclosure. Such a system may be taken to include one or more processors
15 and one or more computer-readable storage mediums. For example, the imaging device 500 and the visual analysis device 600 as described hereinbefore may each include a processor (or controller) and a computer-readable storage medium (or memory) which are for example used in various processing carried out therein as described herein. A memory or computer-readable storage medium used in various embodiments may be a
20 volatile memory, for example a DRAM (Dynamic Random Access Memory) or a non-volatile memory, for example a PROM (Programmable Read Only Memory), an EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM), or a flash memory, e.g., a floating gate memory, a charge trapping memory, an MRAM (Magnetoresistive Random Access Memory) or a PCRAM (Phase Change Random
25 Access Memory).

[0060] In various embodiments, a “circuit” may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, in an embodiment, a “circuit” may be a hard-wired logic circuit or a programmable logic
30 circuit such as a programmable processor, e.g., a microprocessor (e.g., a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A “circuit” may also be a processor executing software, e.g., any kind of computer program, e.g., a computer program using a virtual machine code, e.g., Java. Any other kind of implementation of the respective functions which will be

described in more detail below may also be understood as a “circuit” in accordance with various alternative embodiments. Similarly, a “module” may be a portion of a system according to various embodiments in the present invention and may encompass a “circuit” as above, or may be understood to be any kind of a logic-implementing entity
5 therefrom.

[0061] Some portions of the present disclosure are explicitly or implicitly presented in terms of algorithms and functional or symbolic representations of operations on data within a computer memory. These algorithmic descriptions and functional or symbolic representations are the means used by those skilled in the data processing arts to convey
10 most effectively the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities, such as electrical, magnetic or optical signals capable of being stored, transferred, combined, compared, and otherwise manipulated.

[0062] Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as “obtaining”, “extracting”, “producing” “transmitting”, “receiving”, “producing”, “performing”, “repacking”, “forming”, “quantizing”, “de-repacking”, “de-quantizing” or the like, refer to the actions and processes of a computer system, or
20 similar electronic device, that manipulates and transforms data represented as physical quantities within the computer system into other data similarly represented as physical quantities within the computer system or other information storage, transmission or display devices.

[0063] The present specification also discloses a system (e.g., which may also be embodied as a device or an apparatus) for performing the operations/functions of the methods described herein. Such a system may be specially constructed for the required purposes, or may comprise a general purpose computer or other device selectively activated or reconfigured by a computer program stored in the computer. The algorithms presented herein are not inherently related to any particular computer or
25 other apparatus. Various general-purpose machines may be used with computer programs in accordance with the teachings herein. Alternatively, the construction of more specialized apparatus to perform the required method steps may be appropriate.

[0064] In addition, the present specification also at least implicitly discloses a computer program or software/functional module, in that it would be apparent to the

person skilled in the art that the individual steps of the methods described herein may be put into effect by computer code. The computer program is not intended to be limited to any particular programming language and implementation thereof. It will be appreciated that a variety of programming languages and coding thereof may be used to implement the teachings of the disclosure contained herein. Moreover, the computer program is not intended to be limited to any particular control flow. There are many other variants of the computer program, which can use different control flows without departing from the spirit or scope of the invention. It will be appreciated by a person skilled in the art that various modules described herein (e.g., the sensor data obtaining module 506, the intermediate deep feature extracting module 508, the video data encoding module 510 and/or the encoded video data transmitting module 512 in relation to the imaging device 500, or the encoded video data receiving module 606, the video data decoding module 608, the intermediate deep feature producing module 610 and/or the visual analysis performing module 612 in relation to the visual analysis device 600) may be software module(s) realized by computer program(s) or set(s) of instructions executable by one or more computer processors to perform the required functions, or may be hardware module(s) being functional hardware unit(s) designed to perform the required functions. It will also be appreciated that a combination of hardware and software modules may be implemented.

[0065] Furthermore, one or more of the steps of a computer program/module or method described herein may be performed in parallel rather than sequentially. Such a computer program may be stored on any computer readable medium. The computer readable medium may include storage devices such as magnetic or optical disks, memory chips, or other storage devices suitable for interfacing with a general purpose computer. The computer program when loaded and executed on such a general-purpose computer effectively results in an apparatus that implements the steps of the methods described herein.

[0066] In various embodiments, there is provided a computer program product, embodied in one or more computer-readable storage mediums (non-transitory computer-readable storage medium), comprising instructions (e.g., the sensor data obtaining module 506, the intermediate deep feature extracting module 508, the video data encoding module 510 and/or the encoded video data transmitting module 512) executable by one or more computer processors to perform a method 300 of visual data transmission for network-based visual analysis as described hereinbefore with

reference to FIG. 3. Accordingly, various computer programs or modules described herein may be stored in a computer program product receivable by a system (e.g., which may also be embodied as a device or an apparatus) therein, such as the imaging device 500 as shown in FIG. 5, for execution by at least one processor 504 of the imaging device 500 to perform the required or desired functions.

[0067] In various embodiments, there is provided a computer program product, embodied in one or more computer-readable storage mediums (non-transitory computer-readable storage medium), comprising instructions (e.g., the encoded video data receiving module 606, the video data decoding module 608, the intermediate deep feature producing module 610 and/or the visual analysis performing module 612), executable by one or more computer processors to perform a method 400 of network-based visual analysis as described hereinbefore with reference to FIG. 4. Accordingly, various computer programs or modules described herein may be stored in a computer program product receivable by a system (e.g., which may also be embodied as a device or an apparatus) therein, such as the visual analysis device 600 as shown in FIG. 6, for execution by at least one processor 604 of the imaging device 600 to perform the required or desired functions.

[0068] The software or functional modules described herein may also be implemented as hardware modules. More particularly, in the hardware sense, a module is a functional hardware unit designed for use with other components or modules. For example, a module may be implemented using discrete electronic components, or it can form a portion of an entire electronic circuit such as an Application Specific Integrated Circuit (ASIC). Numerous other possibilities exist. Those skilled in the art will appreciate that the software or functional module(s) described herein can also be implemented as a combination of hardware and software modules.

[0069] In various embodiments, the imaging device 500 may be realized by any device (e.g., which may also be embodied as a system or an apparatus) having an image capturing component or unit (e.g., an image sensor), communication functionality or capability (e.g., wired or wireless communication interface), a memory and at least one processor communicatively coupled to the memory, such as but not limited to, a smartphone, a wearable device (e.g., a smart watch, a head-mounted display (HDM) device, and so on), and a camera (e.g., a portable camera, a surveillance camera, a vehicle or dashboard camera, and so on). By way of an example only and without limitation, the imaging device 500 may be a portable or mobile computing device 700

as schematically shown in FIG. 7. Various methods/steps or functional modules (e.g., the sensor data obtaining module 506, the intermediate deep feature extracting module 508, the video data encoding module 510 and/or the encoded video data transmitting module 512) may be implemented as software, such as a computer program being executed within the portable computing device 700, and instructing the portable computing device 700 (in particular, at least one processor therein) to conduct the methods/functions of various embodiments described herein.

[0070] The portable computing device 700 may comprise a processor module 702, an input module such as a keypad 704 and an output module such as a display screen 706. It can be appreciated by a person skilled in the art that the display screen 706 may be a touch-sensitive display screen, and thus may also constitute an input module being in addition to, or instead of, the keypad 704. That is, it can be appreciated by a person skilled in the art that the keypad 704 may be omitted from the portable computing device 700 as desired or as appropriate. The processor module 702 is coupled to a first communication unit 708 for communication with a cellular network 710. The first communication unit 708 can include but is not limited to a subscriber identity module (SIM) card loading bay. The cellular network 710 can, for example, be a 3G, 4G or 5G network. The processor module 702 may further be coupled to a second communication unit 712 for connection to a local area network 714. For example, the connection can enable wired or wireless communication and/or access to, e.g., the Internet or other network systems such as Local Area Network (LAN), Wireless Personal Area Network (WPAN) or Wide Area Network (WAN). The second communication unit 712 may include but is not limited to a wireless network card or an Ethernet network cable port. The processor module 702 in the example includes a processor 716, a Random Access Memory (RAM) 718 and a Read Only Memory (ROM) 720. The processor module 702 also includes a number of Input/Output (I/O) interfaces, for example I/O interface 722 to the display screen 706, and I/O interface 724 to the keypad 704. The components of the processor module 702 typically communicate via an interconnected bus 726 and in a manner known to the person skilled in the relevant art. Various software or application programs (or may simply be referred to herein as “apps”) may be pre-installed in a memory of the mobile communication device 700 or may be transferred to a memory of the mobile communication device 700 by reading a memory card having

stored therein the application programs or by downloading wirelessly from an application server (e.g., an online app store).

- [0071] In various embodiments, the visual analysis device 600 may be realized by any computer system (e.g., desktop or portable computer system, which may also be embodied as a device or an apparatus) including at least one processor and a memory, such as a computer system 800 as schematically shown in FIG. 8 as an example only and without limitation. Various methods/steps or functional modules (the encoded video data receiving module 606, the video data decoding module 608, the intermediate deep feature producing module 610 and/or the visual analysis performing module 612) may be implemented as software, such as a computer program being executed within the computer system 800, and instructing the computer system 800 (in particular, one or more processors therein) to conduct the methods/functions of various embodiments described herein. The computer system 800 may comprise a computer module 802, input modules, such as a keyboard 804 and a mouse 806, and a plurality of output devices such as a display 808, and a printer 810. The computer module 802 may be connected to a computer network 812 via a suitable transceiver device 814, to enable access to e.g., the Internet or other network systems such as Local Area Network (LAN) or Wide Area Network (WAN). The computer module 802 in the example may include a processor 818 for executing various instructions, a Random Access Memory (RAM) 820 and a Read Only Memory (ROM) 822. The computer module 802 may also include a number of Input/Output (I/O) interfaces, for example I/O interface 824 to the display 808, and I/O interface 826 to the keyboard 804. The components of the computer module 802 typically communicate via an interconnected bus 828 and in a manner known to the person skilled in the relevant art.
- [0072] FIG. 9 depicts a schematic block diagram of a network-based visual analysis system 900 according to various embodiments of the present invention. The network-based visual analysis system 900 comprises one or more imaging devices 500, each imaging device 500 being configured for visual data transmission for network-based visual analysis as described hereinbefore according to various embodiments with reference to FIG. 5; and a visual analysis device 600 for network-based visual analysis configured as described hereinbefore according to various embodiments with reference to FIG. 6, and configured to receive encoded video data from the one or more imaging devices 500, respectively.

[0073] It will be appreciated by a person skilled in the art that the terminology used herein is for the purpose of describing various embodiments only and is not intended to be limiting of the present invention. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises" and/or "comprising," when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

10 [0074] Any reference to an element or a feature herein using a designation such as "first," "second," and so forth does not limit the quantity or order of such elements or features. For example, such designations are used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements can be employed, or that the first element must precede the second element. In addition, a phrase referring to "at least one of" a list of items refers to any single item therein or
15 any combination of two or more items therein.

[0075] In order that the present invention may be readily understood and put into practical effect, various example embodiments of the present invention will be described hereinafter by way of examples only and not limitations. It will be appreciated by a person skilled in the art that the present invention may, however, be embodied in various different forms or configurations and should not be construed as limited to the example embodiments set forth hereinafter. Rather, these example
20 embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the present invention to those skilled in the art.

[0076] With the unprecedented success of deep learning in computer vision tasks, many network-based (e.g., cloud-based) visual analysis applications are powered by deep learning models. However, the deep learning models are also characterized with high computational complexity and may be task-specific, which may hinder the large-scale implementation of the conventional data communication paradigms. To enable a better balance among bandwidth usage, computational load and the generalization capability for cloud-end server(s), various example embodiments provide network-based visual analysis, and more particularly, a method of visual data transmission for network-based visual analysis, that compresses and transmits intermediate deep
30

- learning feature(s) (which may interchangeably be referred to herein as intermediate deep feature(s) or intermediate layer feature(s)) instead of visual signals (i.e., visual data at the signal level, e.g., direct visual signals produced by an image sensor) or ultimately utilized feature(s). The method according to various example embodiments
- 5 also provide a promising way for the standardization of deep feature coding. In this regard, various example embodiments provide a lossy compression framework or method and evaluation metrics for intermediate deep feature compression. Experimental results are also presented to show the effectiveness of the method according to various example embodiments and the feasibility of the data transmission
- 10 strategy or method according to various example embodiments. In various example embodiments, the compression framework (lossy compression framework) and evaluation metrics according to various example embodiments may be adopted or employed in the ongoing AVS (Audio Video Coding Standard Workgroup) - Visual Feature Coding Standard.
- 15 **[0077]** FIG. 10 shows a table (Table 1) comparing various attributes associated with three data transmission strategies or methods, namely, the conventional “compress-then-analyse” method (“Transmit Video Signal”), the “analyze-then-compress” method (“Transmit Ultimate Feature”) and the above-mentioned method of data transmission (“Transmit Intermediate Feature”) according to various example embodiments of the
- 20 present invention. In view of various pros and cons of the two conventional paradigms (e.g., summarized in Table 1 shown in FIG. 10), various example embodiments provide a strategy or method of transmitting intermediate layer features of deep learning models (intermediate deep features) instead of visual signals or ultimate features, which has been found to advantageously achieve a balance among the computing load,
- 25 communication cost and the generalization ability. Various example embodiments note that intermediate deep feature compression has not been well explored in literature, thus problems such as how to efficiently compress intermediate deep features from different layers of different deep models with a unified compression framework and how to evaluate the compression methods are not addressed in the literature.
- 30 **[0078]** In particular, various example embodiments:
- present and analyze the data communication strategy of transmitting intermediate deep features for cloud-based visual analysis applications, which enables a good balance among the transmission load, computing load and the generalization ability for cloud servers;

- provide a video codec based lossy compression framework for intermediate deep feature coding, which can provide good performance and make full use of the video coding infrastructures when upgrading the communication system; and
- 5 • introduce new metrics for fidelity evaluation of intermediate deep feature compression methods, and report comprehensive experimental results.

[0079] A detailed description on the data transmission and compression of intermediate deep features according to various example embodiments of the present invention will now be described. Subsequently, a lossy intermediate deep feature
10 compression framework and the evaluation metrics according to various example embodiments will be described, as well as experimental results on the methods and metrics according to various example embodiments.

TRANSMISSION AND COMPRESSION OF INTERMEDIATE DEEP FEATURES

15 *Intermediate Deep Feature Transmission*

[0080] In the context of network-based (e.g., cloud-based) visual analysis, visual signal acquisition and analysis may be processed in distributed devices. Sensor data (e.g., images and videos) may be captured at the front end (e.g., surveillance cameras and smart phones), while the analyses may be completed in the cloud-end server(s).
20 Conventionally, the data communication between the front and cloud ends can be with either visual signals or ultimate features, as discussed hereinbefore with reference to FIG. 2A or 2B.

[0081] As discussed hereinbefore (e.g., in the background), in relation to transmitting visual signal (i.e., the conventional “compress-then-analyse” approach) as
25 shown as in FIG. 2A, all types of visual analyses, including manual monitoring, are applied at the cloud end, since the image/video data are available. However, due to the visual signal degradation resulting from lossy image/video compression, the performance drop of the analysis tasks is non-negligible, especially when the compression ratio is high. Moreover, it is doubtful that whether such signal level
30 communication can efficiently handle the visual big data, as all the computing load for visual analyses allocates on the cloud-end server(s). In relation to transmitting ultimate feature (i.e., the conventional “analyze-then-compress” approach) as shown in FIG. 2B, the computing load on the cloud side can be largely shifted to the front-end devices,

which makes cloud-based visual analysis feasible in the context of big data. However, as deep learning models are trained in a data-driven manner, the top-layer features (ultimate features) are usually task-specific and hard to generalize to different types of visual analysis tasks. In the conventional “analyze-then-compress” approach, to enable multiple types of analysis in the cloud side, for example, different deep learning models may need to be deployed in the front-end device, which makes the whole system bloated and complicated. In other words, the availability of visual analysis applications in cloud-end servers is unduly or unsatisfactorily constrained by different deep learning models implemented in the front-end devices.

10 **[0082]** FIG. 11 depicts a schematic drawing of a network-based (e.g., cloud-based) visual analysis system 1100 (e.g., corresponding to the network-based visual analysis system 900 as described hereinbefore according to various embodiments) according to various example embodiments of the present invention. The network-based visual analysis system 1100 comprises one or more imaging devices 1104 (at the front end, e.g., each imaging device 1104 corresponding to the imaging device 500 as described hereinbefore according to various embodiments), each imaging device 1104 being configured for visual data transmission for network-based visual analysis; and a visual analysis device 1108 (at the server or cloud end, e.g., corresponding to the visual analysis device 600 as described hereinbefore according to various embodiments) for network-based visual analysis and configured to receive encoded video data from the one or more imaging devices 1104, respectively. In various example embodiments, to balance the computing load between the front and cloud ends without limiting (e.g., without unduly or unsatisfactorily limiting) the analysis capability in the cloud side, the network-based visual analysis system 1100 (in particular, the front end) is configured to transmit the intermediate deep features instead of visual signals and ultimate features. In particular, as shown in FIG. 11, the intermediate deep features of a generic deep model can be applied to a broad range of tasks. In this regard, for example, the intermediate deep features of specific layers may be transmitted based on the analysis requirements on the cloud side. On top of these transmitted features, shallow task-specific models may be applied at the server end for visual analysis. Various example embodiments note that deep neural networks are with hierarchical structures, which can be considered as a combination of cascade feature extractors rather than a single straightforward feature extractor. Various example embodiments note that the intermediate deep features from upper layers of intermediate layers are more abstract

and task-specific, while the intermediate deep features from lower layers of intermediate layers can be applied to a broader range of analysis tasks. As such, according to various example embodiments, the cloud-end server(s) may request any intermediate features as desired or as appropriate from the front end according to the analysis tasks. Accordingly, in various example embodiments, a generic deep model whose features can be applied to different visual analysis tasks may be preferred to be deployed at the front end, while light-weighted task-specific neural networks, which takes the transmitted intermediate features as input, may be implemented at the cloud side to perform various analysis tasks as desired or as appropriate.

10 [0083] According to various example embodiments, by way of examples only and without limitations, various deep learning models may be applied, such as but not limited to, VGGNet and ResNet which are widely adopted as the backbone networks in many computer vision tasks. For example, task-specific networks may be built on top of particular intermediate features of the backbone networks. Such backbone networks can be regarded as generic to deploy in the front end. By way of examples only and without limitations, FIG. 12 shows a table (which may be referred to as Table 2 herein) which summarizes the usability of the intermediate deep features according to various example embodiments. In Table 2, in relation to *conv4* or *pool4* features, “captioning” may refer that disclosed in Gu et al., “Stack-captioning: Coarse-to-fine learning for image captioning”, arXiv preprint arXiv:1709.03376, 2017; “QA” may refer to that disclosed in Fukui et al., “Multimodal compact bilinear pooling for visual question answering and visual grounding”, arXiv preprint arXiv:1606.01847, 2016; and “tracking” may refer to that disclosed in Wang et al, “Visual tracking with fully convolutional networks”, In Proceedings of the IEEE International Conference on Computer Vision. 3119–3127, 2015. In relation to *conv5* or *pool5* features, “captioning” may refer that disclosed in Xu, “Show, attend and tell: Neural image caption generation with visual attention”, In International Conference on Machine Learning. 2048–2057, 2015; “QA” may refer to that disclosed in Lu et al., “Hierarchical question-image co-attention for visual question answering”, In Advances In Neural Information Processing Systems pages 289–297, 2016; “tracking” may refer to that disclosed in Wang et al, “Visual tracking with fully convolutional networks”, In Proceedings of the IEEE International Conference on Computer Vision. 3119–3127, 2015; “detection” may refer to that disclosed in Girshick, “Fast r-cnn”, arXiv preprint arXiv:1504.08083, 2015 or Ren et al., “Faster r-cnn: Towards real-time object detection with region proposal

networks”, In Advances in neural information processing systems, pages 91-99, 2015; and “retrieval” may refer to that disclosed in Lin et al., “Hnlp: Compact deep invariant representations for video matching, localization, and retrieval”, IEEE Transactions on Multimedia 19, 9 (2017), pages 1968-1983. In relation to *fc* (fully connected) features,

5 “detection” may refer to that disclosed in Girshick et al., “Region-based convolutional networks for accurate object detection and segmentation”, IEEE transactions on pattern analysis and machine intelligence 38, 1 (2016), pages 142-158; and “retrieval” may refer to that disclosed in Chandrasekhar et al, “A practical guide to CNNs and Fisher Vectors for image instance retrieval”, Signal Processing 128 (2016), pages 426-439.

10 **[0084]** In particular, in Table 2, various example embodiments note that the computing cost of neural networks may lay on intermediate features from lower intermediate layers, while most of visual applications may utilize intermediate features from upper intermediate layer. Accordingly, this demonstrates that transmitting intermediate features according to various example embodiments can advantageously

15 help shift the majority of computing load, while maintaining the data usability. For example, from Table 2, most of task-specific networks may take high layer features (e.g., *conv4* or higher) from intermediate layers as their input. Since the computing load are mainly laid on low layers in neural networks, it can help saving great computing cost for the server-end with the network-based visual analysis system 1100 according

20 to various example embodiments of the present invention. Thus, the network-based visual analysis system 1100 according to various example embodiments of the present invention can advantageously help to minimize the computing load in the cloud end while maximizing the availability of various analysis applications. Furthermore, the deep neural networks may be developed to more and more generic in the future,

25 resulting in the network-based visual analysis system 1100 having more advantages over the conventional network-based visual analysis systems, such as those shown in FIGs. 2A and 2B.

Intermediate Deep Feature Compression

30 **[0085]** As discussed hereinbefore, conveying intermediate deep features according to various example embodiments of the present invention, instead of visual signals and ultimate features, is found to be advantageous at reducing the computing load at the cloud end, while maintaining the availability of various or different visual analysis applications. In this regard, various example embodiments further note that the

transmission load for intermediate deep features is non-negligible, and provide compression methods for intermediate deep features.

[0086] By investigating in successful neural network architectures (backbone architectures), such as but not limited to, AlexNet, VGGNet, ResNet and DenseNet, various example embodiments note or found that such network architectures share similar block-wise structures and feature shapes. For example, in convolutional neural networks (CNNs), intermediate deep features are mainly in forms of feature maps which are the combinations of stacked two-dimensional (2D) matrices. The height and width of the feature maps may gradually get reduced along with the inference process. For example, one or several layers can be composed as a block which halves the height and width of the feature maps. So, with the same input size, certain blocks of different network architectures shall provide the feature maps with identical height and width. Furthermore, the numerical distributions of intermediate deep features also share similar properties, as most CNN architectures use ReLU as the non-linear transformation function which clips the features into same numerical range. In view of such observations, according to various example embodiments, intermediate deep features of different network architectures may be compressed with an unified compression method.

20 *Standardization of Intermediate Deep Feature Compression*

[0087] According to various example embodiments of the present invention, intermediate deep feature coding may be standardized to facilitate the data communication of intermediate deep feature in network-based (e.g., cloud-based) visual analysis applications.

25 [0088] Various example embodiments note that feature coding standards, such as CDVS (compact descriptors for visual search) and CDVA (compact descriptors for video analysis), should specify both feature extraction and compression processes to fully ensure the interoperability, as features from different extractors may be with different shape, distribution and numerical type. With such standardization strategy, feature extractors may be carefully designed and specified, which ensures the interoperability but sacrifices the compatibility for different feature extractors and the generality for different tasks. For intermediate deep feature coding, as discussed hereinbefore under the sub-heading “intermediate deep feature compression”, various example embodiments note that features from different deep learning models (feature

extractors) share similar shapes and distributions, which make it possible to obtain the interoperability by only specifying the compression process. Since the choice of deep learning models are left open, the compatibility and generality of the standard can also be ensured together with the interoperability. Moreover, such standardization strategy is also good for keeping the standard with long-lasting vitality, as any new deep neural network with better performance in the future can be seamlessly adopted for system customization.

COMPRESSION AND EVALUATION METHODS

10 [0089] Various example embodiments provide lossy compression for intermediate deep features.

Video Codec based Lossy Compression

15 [0090] In CNNs, the intermediate features are mainly in the form of feature maps which are the combinations of stacked 2D arrays with spatial correlations among the elements, such as shown in FIG. 13. In particular, by way of an example only and without limitations, FIG. 13 depicts visualized feature maps of VGGNet. In various example embodiments, a single channel 2D feature map may be considered or referred to as a frame (one frame), while an intermediate deep feature may be considered or referred to as a video sequence (one video sequence). For example, in FIG. 13, under *conv1*, the three example images shown may correspond to three feature maps of an intermediate deep feature extracted from that intermediate layer, whereby each feature map may be considered as a channel of the intermediate deep feature. That is, under *conv1*, the intermediate deep feature extracted from that intermediate layer comprises the three feature maps, and similarly for the example images shown under other intermediate layers shown in FIG. 13. For example, for one input image, each intermediate layer may be able to output one intermediate deep feature, and the encoding process 1404 according to various example embodiments may process one intermediate deep feature at a time. In various example embodiments, the visual analysis device at the server side may decide which intermediate deep feature (i.e., from which intermediate layer) to select or process depending on various factors, such as the visual analysis task and the computing/communication costs. As such, various example embodiments advantageously apply existing video codecs to compress deep features in a lossy manner. In particular, various example embodiments provide a video codec

based compression framework for intermediate deep feature coding. By integrating video codecs into the compression framework according to various example embodiments, matured video coding techniques can be borrowed or employed to intermediate feature coding seamlessly. Furthermore, as video encoding/decoding modules (e.g., chips, IP cores, and so on) have already been widely deployed in many cloud based systems, it is economically and technically friendly to upgrade or modify the visual devices and systems to support intermediate deep feature conveyance and analysis with the compression framework according to various example embodiments.

[0091] FIG. 14A depicts a schematic flow diagram of network-based visual analysis 1400 (e.g., corresponding to the network-based visual analysis as described hereinbefore according to various embodiments) according to various example embodiments of the present invention, and more particular, a method 1404 of visual data transmission for network-based visual analysis (e.g., corresponding to “encoding process” shown in FIG. 14A and corresponding to the method 300 of visual data transmission for network-based visual analysis as described hereinbefore according to various embodiments) and a method 1408 of network-based visual analysis (e.g., corresponding to “decoding process” shown in FIG. 14A and corresponding to the method 400 of network-based visual analysis as described hereinbefore according to various embodiments), according to various example embodiments of the present invention. FIG. 14B also shows a schematic flow diagram of the network-based visual analysis 1400 according to various example embodiments of the present invention, which is the same as that shown in FIG. 14A but with additional schematic illustrations.

Encoding Process (or Encoding Module)

[0092] In the encoding process or phase (or encoding module) 1404, FIG. 14A shows a schematic flow diagram of a lossy compression method for intermediate deep feature maps according to various example embodiments of the present invention. As shown, in the encoding phase 1404, a pre-quantization operation or step (or pre-quantization module) 1420 (which may also simply be referred to as quantization, e.g., corresponding to the “quantizing the plurality of feature maps to obtain a plurality of quantized feature maps” as described hereinbefore according to various embodiments) may be applied on a plurality of feature maps. In this regard, various example embodiments note that the numerical type of feature maps (or deep features) may not be compatible with the input of video codecs. For example, the vanilla VGGNets and

ResNets features may be in *float32* (i.e., in a floating point format), while video codecs, such as HEVC, are designed for integer input with 8 or higher bit depth. Accordingly, the pre-quantization operation 1420 may be performed to convert the plurality of feature maps (e.g., in a floating point format) to a plurality of quantized feature maps (e.g., in an integer format), respectively. In various example embodiments, different quantizers may be applied based on the distribution analyses of intermediate features.

[0093] After quantization, a repack operation or step (or repack module) 1424 (which may also simply be referred to as packing or organising, e.g., corresponding to the “repacking the plurality of feature maps based on a repacking technique” as described hereinbefore according to various embodiments) may be applied to produce a video format data. For example, after quantization, N feature maps (or N quantized feature maps) $\mathbb{R}^{N \times H \times W \times C}$ may be repacked into a video-sequence-like format (or video format data) $\mathbb{R}^{N' \times W' \times W \times NC}$ to fit the video codec input, where H and W are the height and width of the feature map, C is the channel number (i.e., the number of feature maps) of the feature sample. As the input frame size of the video codec is usually non-arbitrary, such as the input size of HEVC can only be integral multiple of 8, the original feature map size $H \times W$ may be extended to $H' \times W'$ by padding methods. Particularly, $H' = \lceil H/8 \rceil \times 8$ and $W' = \lceil W/8 \rceil \times 8$, where $\lceil \cdot \rceil$ is the ceiling operation. In various example embodiments, the order of the frames may be further reorganized during the repack phase, which may affect the compression performance if inter-frame correlations are considered. Accordingly, as an example, the repacked feature maps may then be considered as 4:0:0 video sequences (which may be a grayscale video, whereby each frame of the video only includes one channel of the repacked feature, and each channel of the repacked feature may be considered as a frame of the video sequences) to feed into the video encoder 1428.

Decoding Process (or Decoding Module)

[0094] In the decoding process or phase (or decoding module) 1408, as shown in FIG. 14A, the received bitstream (e.g., corresponding to the encoded video data as described hereinbefore according to various embodiments) may first be decoded by a video decoder 1408 (e.g., corresponding to the video encoder 1428) to produce decoded video data. Then, a de-repack operation or step (or de-repacking module) 1440 (e.g., which may simply be referred to as de-pack, e.g., corresponding to the “de-repacking

the video format data based on a de-repacking technique” as described hereinbefore according to various embodiments) may be performed to convert the reconstructed video sequence-like data (the decoded video data comprising video format data, the video format data comprising one or more repacked feature maps) to the original feature size (e.g., a plurality of de-repacked feature maps). Subsequently, a de-quantization operation or step (or de-quantization module) 1444 may be performed to de-quantize the plurality of de-repacked feature maps (e.g., integer feature tensors) to float type (e.g., to a plurality of de-quantized feature maps comprising in a floating point format). The plurality of de-quantized feature maps may then constitute a plurality of reconstructed deep feature maps 1448 which may then be passed to task-specific models to perform visual analyses.

[0095] For better understanding, the encoding phase or process 1404 and the decoding phase or process 1408 will now be described in further detail according to various example embodiments of the present invention. FIG. 14A illustrates a hybrid coding framework according to various example embodiments, which integrates traditional video codecs, that can seamlessly borrow matured video coding techniques to help feature maps compression. Moreover, as video codecs are broadly deployed in existing visual analysis systems, both software and hardware development for the hybrid coding framework according to various example embodiments may be readily implemented.

[0096] As shown in FIGs. 14A, the encoding phase 1404 may involve three modules to encode the feature maps to produce an encoded video data. In various example embodiments, the pre-quantization module 1420 and repack module 1424 may transform the feature maps into YUV format data (video format data). After that, a video encoder 1428 (e.g., an appropriate conventional video encoder known in the art) may be applied to compress the YUV format data to produce the encoded video data. With such workflow, as the video encoder 1428 may be developed and specified in advance, the coding performance may largely depend on how the feature data representation can fit the video codec. In view of this, pre-quantization and repack modules may be configured accordingly.

[0097] In various example embodiments, let an intermediate deep learning feature $D \in \mathbb{R}^{W \times H \times C}$ comprise a plurality of 2D arrays $\in \mathbb{R}^{W \times H}$ (i.e., a plurality of feature maps). In this regard, the intermediate deep learning feature D may be referred to as

having C channels, $\mathbb{R}$ denotes a set of real numbers, and $W \times H \times C$ may define the shape of the intermediate deep learning feature.

[0098] In various example embodiments, the pre-quantization operation 1420 may be performed based on a uniform quantization technique, a logarithmic quantization technique or a learning-based adaptive quantization technique (e.g., may be referred to as coding tools or modes). In various example embodiments, the repacking operation 1424 may be performed based on a naive channel concatenation technique, a channel concatenation by distance technique or a channel tiling technique (e.g., may be referred to as coding tools or modes). These quantization techniques and repacking techniques will now be described further below according to various example embodiments of the present invention.

Pre-Quantization

[0099] Various example embodiments note that deep neural networks may be in floating point format with high bit depth to ensure accurate back propagation during training. On the other hand, in inference phase, the output result of a neural network may be insensitive to small changes of intermediate features. In this regard, various example embodiments may perform pre-quantization 1420 to reduce the volume of feature maps. Furthermore, various example embodiments may also perform pre-quantization 1420 to convert the numeric type of feature maps to meet the input requirement of a video codec, such as from a floating point format to an integer format. In this regard, the pre-quantization operation 1420 may be performed to convert the input intermediate deep learning feature D to an integer format with lower (or equal) bit depth, while the shape of the feature may remain the same. The pre-quantization operation 1420 may then output a quantized feature $D_{quant} \in \mathbb{N}_0^{W \times H \times C}$, where $\mathbb{N}_0$ denotes a set of non-negative integers.

[00100] In various example embodiments, any scalar quantization method may be applied as appropriate or as desired. In this regard, a scalar quantization may be a process that maps each input within a specified range to a common or predetermined value. Accordingly, the process may map different inputs in different ranges of values to different common or predetermined values, respectively. By way of example only and without limitations, the above-mentioned uniform quantization technique,

logarithmic quantization technique and learning-based adaptive quantization technique will now be described in further detail below.

[00101] *Uniform quantization*: Various example embodiments may provide a uniform quantization technique configured to evenly sample the activations of feature maps, which may be expressed as, by way of an example only and without limitation:

$$D_{quant} = rint\left(\frac{D - \min(D)}{\max(D) - \min(D)} \cdot (2^{bitdepth} - 1)\right) \quad \text{Equation (1)}$$

where D denotes the original feature maps with high bit depth, D_{quant} is the quantized features. $rint(\cdot)$ rounds the floating point input to the nearest integer.

[00102] *Logarithmic quantization*: Considering the distribution of feature maps which usually has a right-skewed exponential behaviour as shown in FIGs. 15A to 15D, various example embodiments may provide a logarithmic quantization technique (or a logarithmic quantizer with logarithmic sampling methods), which may achieve better performance than the uniform quantizer. By way of an example only and without limitation, the logarithmic quantizer may be expressed as:

$$D_{quant} = rint\left(\frac{\log(D - \min(D) + 1)}{\max(\log(D - \min(D) + 1))} \cdot (2^{bitdepth} - 1)\right) \quad \text{Equation (2)}$$

where $\log(\cdot)$ is a logarithm function with an arbitrary base.

[00103] *Learning based adaptive quantization*: Although distributions shown in FIGs. 15A to 15D show exponential behaviour, various example embodiments note that the exponential functions may not perfectly fit the probability distributions of feature maps data. To more precisely describe the distribution, in various example embodiments, a learning-based quantizer may be provided or applied, which is configured to learn the probability function from massive feature data.

Repack

[00104] In the repack operation or step (or repack module) 1424, the plurality of quantized feature maps (or quantized 2D arrays) may be reorganized to YUV format data (video format data) to feed the subsequent video codec. According to various example embodiments, the repack operation may be configured to enable or facilitate the video encoder 1428 better eliminate redundancy.

[00105] In various example embodiments, the repack operation 1424 may be configured to reorganize the quantized feature data (e.g., a plurality of quantized feature maps) $D_{quant} \in \mathbb{N}_0^{W \times H \times C}$ to one or more repacked feature maps $D_{repack} \in$

$\mathbb{N}_0^{W' \times H' \times C'}$, to help the subsequent video codec better explore and eliminate the redundancy in feature data. In the repacking operation 1424, the values and numerical type of elements of the feature data D_{quant} may not be changed. However, the shape of feature data and the indices of the elements may change. In various example
 5 embodiments, the operation of “reorganizing” the feature data may include (a) mapping elements of D_{quant} to D_{repack} (i.e., change indices of elements of the feature data), and (b) inserting new elements to the repacked feature D_{repack} . Accordingly, the element number of D_{quant} (i.e., $W \times H \times C$) may not necessarily be the same as the element number of D_{repack} (i.e., $W' \times H' \times C'$). By way of examples only and without
 10 limitations, the above-mentioned naive channel concatenation technique, channel concatenation by distance technique and channel tiling technique will now be described in further detail below.

[00106] *Naive channel concatenation:* A naïve simple or approach may be to repack the feature maps $\mathbb{N}_0^{W \times H \times C}$ by simply concatenating all the channels. In this way, each
 15 channel $\mathbb{N}_0^{H \times W}$ may be considered as a gray-scale frame, while the entire C channels may be composed as a video sequence. As typical spatial correlations in each channel are rich, intra-channel redundancy can be neatly identified by intra prediction tools in traditional video codecs. However, in contrast to video signals, there do not exist explicit motion among channels of feature maps. Existing inter prediction techniques,
 20 such as motion estimation, may not be effective for redundancy elimination among channels.

[00107] *Channel concatenation by distance:* To enable better performance for inter-channel redundancy elimination, various example embodiments improve the naïve channel concatenation technique by reorganizing the order of feature channels to
 25 minimize the distance of nearby feature maps (e.g., immediately adjacent pair of feature maps), such as described in Algorithm 1 shown in FIG. 16. In various example embodiments, L2 norm may used to calculate the distance between channels (e.g., an inter-channel distance between an adjacent pair of feature maps). With such an approach, residual information between nearby channels is decreased, thereby
 30 escalating compression ratio.

[00108] Accordingly, in the above-mentioned naive channel concatenation technique and the above-mentioned channel concatenation by distance technique, the feature maps (e.g., 2D arrays) may be concatenated along the channel dimension. In the

case of the naïve channel concatenation technique, the order of the channels (feature maps) in D_{repack} may be maintained the same as in D_{quant} . In the case of the channel concatenation by distance technique, such as illustrated in FIG. 17A, the order of the channels (feature maps) in D_{repack} may be determined based on inter-channel distances (e.g., Euclidean distance) associated with the channels.

[00109] Accordingly, in various example embodiments, with the above-mentioned channel concatenation techniques, the index of the element ($D_{\text{quant}}[w, h, c]$) in the feature map (or feature data) may only be changed for its 'C' axis. In this regard, repacking supplemental information (e.g., indexing information) may be generated indicating a mapping relationship between the plurality of quantized feature maps and the plurality of repacked feature maps, such as in the form of a list of indices that sort D_{quant} into D_{repack} along the C axis. Accordingly, as will be described later in the de-repacking operation (i.e., inverse of the repacking operation), D_{repack} may be inverted or restored to D_{quant} , based on the indexing information.

[00110] *Channel tiling*: Various example embodiments provide a channel tiling technique to facilitate the video codec to identify the inter-channel redundancy by tiling the channels (feature maps). In this technique, for example, one channel of the feature (i.e., one feature map) may be considered as one patch of a frame, rather than an entire frame. By way of an example only and without limitation, FIG. 17B illustrates an example channel tiling technique according to various example embodiments. As shown in FIG. 17B, the channel tiling technique may be configured to compose the feature maps (2D arrays) into one or more enlarged feature maps (enlarged 2D array). In this regard, each enlarged feature map may be considered as or may constitute one frame in the input video sequence for the subsequent video codec. Inter-channel redundancy of feature maps may then be subsequently be explored by intra coding tools of the subsequent video codec.

[00111] After channel re-organizing (e.g., channel concatenation or channel tiling) by a repacking technique as described above, the plurality of repacked feature maps (which may also be referred to as a three-dimensional (3D) array) may constitute video format data (e.g., YUV400 format, that is, 4:0:0 video sequences which may be a grayscale video) as an input to the subsequent video encoder 1428. In various example embodiments, the height and width of the 3D array may be extended to integral multiple of 8 with a replicate padding method.

Video Encoder

[00112] In various example embodiments, the repacked YUV data (video format data) may be encoded by a video encoder 1428 using a conventional video codec. It will be understood by a person skilled in the art that any video codec known in the art may be employed as desired or as appropriate. By way of an example only and without limitation, HEVC (High-Efficiency Video Coding) may be employed and is used to conduct various experiments discussed herein.

10 Decoding process

[00113] In various example embodiments, in relation to the network-based visual analysis 1400, the decoding process or phase 1408 corresponds (inversely) to the encoding process or phase 1404 as described hereinbefore according to various example embodiments, therefore, various functions or operations (e.g., stages) configured to be performed by the decoding process 1408 may correspond (inversely) to various functions or operations of the encoding process 1404, and thus need not be repeated with respect to the encoding process 1404 for clarity and conciseness. In other words, various example embodiments described herein in context of the encoding process 1404 are analogously valid (inversely) for the corresponding decoding process 1408, and vice versa. Accordingly, in various example embodiments, as shown in FIG. 14A, the decoding process 1408 may include a video decoding operation 1436 corresponding (inversely) to the video encoding operation 1428, a de-repacking operation 1440 corresponding (inversely) to the repacking operation 1424, and a re-dequantization operation 1444 corresponding (inversely) to the pre-quantization operation 1420. For illustration purpose by way of examples only, the decoding process 1408 will be described in further detail below.

[00114] In the decoding process or phase 1408, after the encoded video data received has been decoded by the video decoder 1436 using a video codec, the decoded video data (comprising video format data including one or more repacked feature maps repacked by the repacking operation 1424), such as in the form of $D'_{\text{repack}} \in \mathbb{N}_0^{W_I \times H_I \times C_I}$, may be input to the de-repacking operation 1440, which may have the same shape and numerical type as D_{repack} produced by the repacking operation 1424. D_{repack} corresponds to (e.g., the same as) D'_{repack} . After the de-repacking operation

1440, the plurality of de-repacked feature maps may be input to the de-prequantization operation 1440 for producing a plurality of de-quantized feature maps, such as in the form of $D'_{quant} \in \mathbb{N}_0^{W \times H \times C}$. Similarly, D'_{quant} may have the same shape and numerical type as D_{quant} produced by the pre-quantization operation 1420. The plurality of de-quantized feature maps may thus result in (e.g., constitute) an intermediate deep feature (i.e., a reconstructed intermediate deep feature), such as in the form of $D' \in \mathbb{R}^{W \times H \times C}$, which corresponds to (e.g., same as) the original intermediate deep feature D in the encoding process 1404. Accordingly, similarly, the reconstructed intermediate deep feature D' may have the same shape and numerical type as the original intermediate deep feature D .

De-Repack

[00115] The de-repacking operation 1440 may be configured to de-repack (or reconstruct) the decoded video data (comprising video format data including one or more repacked feature maps repacked by the repacking operation 1424), such as in the form of D'_{repack} , into a plurality of de-repacked feature maps, which corresponds to the plurality of quantized feature maps produced by the pre-quantization operation 1420, such as in the form of D'_{quant} , based on repacking supplemental information (e.g., metadata). The repacking supplemental information or data may comprise indexing information indicating a mapping relationship between the plurality of quantized feature maps D_{quant} produced by the quantizing operation 1420 and the plurality of repacked feature maps D_{repack} produced by the repacking operation 1424. The indexing information thus also indicates a mapping relationship between one or more repacked feature maps D'_{repack} in the decoded video data from the video decoder 1436 and the plurality of de-repacked feature maps, corresponding to the plurality of quantized feature maps, D'_{quant} to be produced by the de-repacking operation 1440, such as for each element of D'_{quant} correspondingly in D'_{repack} . In various example embodiments, the repacking supplemental information may be transmitted to server end along with the bitstream (including the encoded video data) from the front end or may be predetermined at the server end. For example, in the case of channel de-concatenation, the repacking supplemental data may include a list of indices that sort D_{quant} along the C axis. For example, in the case of channel de-tiling, the repacking

supplemental data may include the number of feature maps in height and width axis respectively in the enlarged feature map (2D) array.

De-PreQuantization

- 5 [00116] The de-prequantization operation or module 1444 may be configured to de-quantize the plurality of de-repacked feature maps $D'_{quant} \in \mathbb{N}_0^{W \times H \times C}$ from the de-repacking operation 1440 to obtain a plurality of de-quantized feature maps $D' \in \mathbb{R}^{W \times H \times C}$, respectively. As described in the pre-quantization operation 1420, scalar quantization may be applied in the encoding process 1404. Accordingly (i.e.,
- 10 correspondingly), to dequantize D'_{quant} , quantization supplemental information (e.g., quantization metadata) which is configured to derive the partition and codebook of the quantization process may be utilized. For example, in the case of uniform quantization and logarithmic quantization, the quantization supplemental data may include the bit depth number of D_{quant} and the maximum value of D . For example, in the case of
- 15 learning-based adaptive quantization, the quantization supplemental information may include a vector of partition. In various example embodiments, similarly, the quantization supplemental information may be transmitted to server end along with the bitstream (including the encoded video data) from the front end or may be predetermined at the server end.

20

Evaluation Metrics

- [00117] Similar to video coding, according to various example embodiments, the evaluation of intermediate deep feature coding take both compression performance and information loss into consideration. In various example embodiments, compression rate
- 25 is employed to evaluate the compression performance, which is defined as:

$$\text{Compression rate} = \frac{\text{data volume after compression}}{\text{data volume before compression}}$$

(Equation 4)

- [00118] In various example embodiments, to evaluate information loss, the comparison of output results of the tasks performed after the feature transmission is
- 30 considered. This is because signal-level comparison (e.g., SNR, PSNR) for features is bootless, as deep features are with high level semantic information. It may also be not proper to utilize task performance metrics (e.g., accuracy for image classification task,

mAP for image retrieval task) to evaluate the performance of feature codecs. The reason may for example be threefold. Firstly, the variation of a task performance metric may not reflect the fidelity level of the features before/after compression. Concretely, in terms of the direction of change, information loss of the features before/after compression can result in either positive or negative change to a task performance metric (e.g., classification accuracy varies from 0.80 to 0.75 or 0.85); in terms of the amount of change, same change amount of a task performance metric may refer to different information loss levels. The task performance metric may not be linearly proportional to information loss. Secondly, it may not be well normalized to use task performance metrics to evaluate information loss. On the one hand, task performance metrics are with different value ranges (e.g., image classification accuracy is with the range of 0 to 1, while image captioning CIDEr (e.g., as disclosed in Vedantam et al., “Cider: Consensus-based image description evaluation”, In CVPR, 2015) can reach more than 1); on the other hand, the task performance value on pristine features (i.e., the reference value) may vary depending on the test dataset, which makes it hard to compare information loss with task performance metrics. Thirdly, using the task performance metric to evaluate the information loss, paired values (before/after compression) may be involved which is not elegant.

[00119] Accordingly, various example embodiments provide or configure new metrics to evaluate information loss of features on different tasks. In various example embodiments, three popular computer vision tasks in surveillance applications are selected, namely, image classification, image retrieval and image object detection, respectively. For image classification, various example embodiments calculate the fidelity by comparing the pristine classification DNN outputs (i.e., the onehot classification results) with the outputs inferred from the reconstructed intermediate deep features, as below:

$$F_{classification} = 1 - \frac{1}{2N} \sum_i^N \frac{Hamming(Y_p^i, Y_r^i)}{length(Y_p^i)}$$

(Equation 5)

where Y_p^i is the pristine onehot output of the tested neural network inferred from i -th test image sample, Y_r^i is the onehot output inferred from the corresponding reconstructed intermediate feature, $length(\cdot)$ returns the dimension of input, N denotes the total number of tested samples.

[00120] For retrieval task, given a query, a ranked sequence of documents will be returned by the system. In task performance metrics such as mean average precision (mAP), the order of the ranked sequence is taken into consideration to calculate the average precision (AP). In various example embodiments, the fidelity is calculated by comparing the pristine output document sequence with the one inferred from the reconstructed intermediate deep features:

$$F_{retrieval} = \frac{1}{N} \sum_i^N bubble_index(S_p^i, S_r^i)$$

(Equation 6)

where S_p^i and S_r^i are the ranked sequence of documents returned by the retrieval system with pristine features and reconstructed respectively for the i -th query, N denotes the total number of tested queries, $bubble_index(\cdot, \cdot)$ is provided or configured to measure the similarity between two ranked sequences by counting the number of swap operations during sorting the reconstructed sequence into the pristine with bubble sort method. The similarity measurement after “bubble sort” method may be referred to as “bubble index”. The workflow of bubble index is described in Algorithm 2 shown in FIG. 18. It is worth noting that a naïve implementation of bubble index is computational heavy ($\mathcal{O}(n^2)$) especially when the length of input sequence is large. The computing complexity can be significantly reduced (less than ($\mathcal{O}(n \log(n))$)) by applying dichotomy in the for-loop. The code implementation can be found in Bojarski et al., “End to end learning for self-driving cars”, arXiv preprint, arXiv:1604.07316 (2016).

[00121] As to object detection task, the detection model predicts both location and category of detected objects. We use Intersection over Union (IoU) to measure the fidelity on the location of the predictions, and a relative change rate to monitor the predicted classification confidences. Moreover, considering that predictions with different confidence level contribute differently to the task performance, we weighted each prediction with the confidence inferred by the pristine feature. Overall, the fidelity in object detection task is calculated as below:

$$F_{detection} = \frac{1}{N} \sum_i^N \frac{\sum_j^M UoI(B_p^{ij}, B_r^{ij}) \cdot \max(0, 1 - \frac{C_p^{ij} - C_r^{ij}}{C_p^{ij}}) \cdot C_p^{ij}}{\sum_j^M C_p^{ij}}$$

(Equation 7)

where B is the predicted bounding box and C is the confidence value of the predicted category, N is the number of tested images and M is the number of predicted objects of i -th image. The implement code can be found at the above-mentioned Bojarski document.

5

EXPERIMENTAL RESULTS

[00122] To demonstrate the feasibility of the method of transmitting intermediate deep feature and the effectiveness of the lossy compression framework according to various example embodiments of the present invention, experiments on intermediate
10 deep feature compression on three widely-used visual surveillance tasks with two commonly-used backbone neural networks were conducted and the experimental results presented below.

Experiment Setup

15 [00123] *Evaluation tasks and datasets.* As discussed hereinbefore under the section “Toward Transmission and Compression of Intermediate Deep Features”, an advantage of the data transmission strategy or method according to various example embodiments relies on that intermediate deep features are with good generic ability which can be applied to a broad range of tasks. As such, in experiments conducted, various example
20 embodiments compress the intermediate features from unified backbone networks, and then evaluate the information loss on three notable tasks in visual surveillance, namely, image classification, image retrieval and image object detection, respectively.

[00124] *Image classification:* As a fundamental task in computer vision, image classification has been widely adopted in training and evaluating deep learning
25 architectures. Many generic networks trained on image classification (e.g., VGGNet, ResNet) are employed as feature extractors or backbone networks in other computer vision tasks. Information loss in feature compression on image classification task was evaluated with a subset of the validation set of the ImageNet 2012 dataset (e.g., as disclosed in Russakovsky et al., “Imagenet large scale visual recognition challenge”,
30 International Journal of Computer Vision 115, 3 (2015), pages 211 - 252. To economize the compression time while maintaining the variety of test image categories, one image was randomly chosen from each of the 1,000 classes.

[00125] *Image retrieval:* Content-based image retrieval is another key problem in computer vision. Among image retrieval problems, vehicle retrieval, as a unique

application, has been drawing more and more attention due to the explosively growing requirement on surveillance security field. In the experiments, the “Small” test split of PKU VehicleID dataset (e.g., as disclosed in Liu *et al.*, “Deep relative distance learning: Tell the difference between similar vehicles”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2167-2175, 2016) was adopted to perform feature compression evaluation on image retrieval task, which contains 800 query images and 5,693 reference images. In the experiments, only the features extracted from query images are to be compressed. Features extracted from reference images serve as reference during fidelity evaluations.

- 5
- 10 **[00126] Image object detection:** Image object detection task predicts both object location and category in the mean time, which contains both regression and classification. It is a fundamental task for surveillance analyses. The compression algorithm according to various example embodiments was tested on image object detection with the test set of Pascal Visual Object Classes (VOC) 2007 dataset
- 15 (Everingham *et al.*, “The PASCAL Visual Object Classes Challenge 2007 (VOC 2007) Results”, 2007, which contains 4,952 images and 12,032 objects.

- [00127] Deep learning architectures and features.** In the experiments, intermediate deep features were extracted with VGGNets and ResNets, which are the common choices for image feature extraction in many computer vision applications as their
- 20 features can be regarded as generic.

- [00128] VGGNet:** Simonyan and Zisserman developed VGGNet at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014. VGGNet-16 stands out from the six variants of VGGNet for its good balance among performance and computing complexity. VGG-16 is very appealing thanks its neat architecture
- 25 consisting of 16 convolutional layers which only performs 3×3 convolution and 2×2 pooling all the way through. Currently, it is the most preferred choice to extract features from images in computer vision community. In the experiments, conv1 to pool5 features were extracted from VGGNet-16 architecture to compress and evaluate in image classification; pool3 and pool4 features were not included in image retrieval task
- 30 due to feature downsampling in conv1 and conv2 by setting convolution stride instead of pool3 and pool4 features; pool5 feature was not included in detection task due to the region proposal network (RPN) of faster RCNN is built on top of conv5 feature of VGGNet. The implementation of image classification follows Simonyan et al., ILSVRC-2014 model (VGG team) with 16 weight layers, image retrieval follows Lou

et al., “Embedding Adversarial Learning for Vehicle Re-Identification”, IEEE Transactions on Image Processing, 2019, and image object detection follows Chen *et al.*, “An Implementation of Faster RCNN with Study for Region Sampling”, arXiv preprint arXiv:1702.02138, 2017.

5 **[00129]** *ResNet*: At the ILSVRC 2015, He *et al.* introduced Residual Neural Network (ResNet) (e.g., as disclosed in He *et al.*, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770-778, 2016), which contains a novel technique called “skip connections”. Due to this new structure, the
 10 network architectures are able to go into very deep with lower complexity than VGGNet. ResNets have three commonly used variants with 50, 101, 152 layers respectively. In various example embodiments, the *conv1* to *conv5* and *pool1* features (ResNets do not have pooling layers for the last four blocks) are investigated in image classification and retrieval tasks, the *conv1* to *conv4* and *pool1* (RPN of faster RCNN is built on top of
 15 *conv4* feature of ResNets, so *conv5* feature is not included herein) features are involved in image object detection task. To broadly investigate the features of three variants of ResNets while economizing the implementation difficulty, ResNet-152 was applied for image classification following Kaiming et al., “Deep Residual Learning for Image Recognition”, ResNet-50 for image retrieval following Yihang et al., “Embedding
 20 Adversarial Learning for Vehicle Re-Identification”, IEEE Transactions on Image Processing (2019), and ResNet-101 for image object detection following Chen *et al.*, “An Implementation of Faster RCNN with Study for Region Sampling”, arXiv preprint arXiv:1702.02138 (2017).

[00130] *Configurations for compression*. The video codec based lossy compression
 25 framework as described hereinbefore in section “Video codec based lossy compression” was applied in the experiments. Specifically, for the pre-quantization and pre-dequantization modules (which may simply be referred to as quantization and dequantization modules, respectively), the intermediate deep features are quantized/dequantized with a simple logarithmic sampling method:

$$30 \quad X_{quant} = round \left(\frac{\log(X - \min(X) + 1)}{\max(\log(X - \min(X) + 1))} \cdot (2^8 - 1) \right)$$

(Equation 7)

$$X_{dequant} = 2^{\frac{x_{quant} \cdot \max(\log_2(X - \min(X) + 1))}{2^8 - 1}} + \min(X) + 1.$$

(Equation 8)

[00131] For the repack module 1424, the size of feature maps is extended to be the integral multiple of 8 by padding after the last array element along each dimension with repeating border elements. The order of the feature map channels are kept the same, as only intra coding will be applied subsequently. As to the video encoder/decoder modules 1428/1436, the reference software (HM16.12) of HEVC Range extension (RExt) was employed in the experiments. The compression is performed with four quantization parameter (QP) values, i.e., [12, 22, 32, 42].

Experimental Results

[00132] In the experiments, the intermediate deep features were firstly extracted by neural networks, then passed to the feature encoder to generate compact bitstreams. The compression rate was subsequently calculated with the volume of original intermediate deep features and the corresponding bitstreams by Equation (4). As to the fidelity evaluation, the reconstructed features were passed to their birth-layer of the corresponding neural network to infer the network outputs, which were then compared with pristine outputs to evaluate the information loss of the lossy compression methods by the new metrics described in section “*Evaluation Metrics*”. The exhaustive results are listed in Table 3 in FIG. 19.

[00133] Comparing with lossless compression results reported in, Zhuo et al., “Intermediate deep feature compression: the next battlefield of intelligent sensing”, arXiv preprint arXiv:1809.06196 (2018), it can be observed that lossy deep feature compression methods have more potential to compress the feature data into smaller volume than the lossless methods. In the extreme case, i.e., ResNet *conv4* feature on retrieval dataset can reach over 500 times compression ratio at QP44, while the lossless methods can only reach 2-5 times. However, greater compression ratio results in greater information loss. For each feature type, the fidelity value decreases as the QP value rises. Looking into Table 3, it can be also observed that QP22 can generally provide high fidelity and fair compression ratio in the meantime. Moreover, upper layer features, such as *conv4* to *pool5*, are generally more robust to heavy compression. It is a great character for practical implementation of intermediate feature transmission, since the

high layer features can largely save the computing load, while provide great usability at the cloud end, as mentioned in Table 2 in FIG. 12.

FURTHER EXPERIMENTAL RESULTS

- 5 **[00134]** To validate the effectiveness of the repack module 1424 and the pre-quantization module 1420, feature maps compression experiments were conducted with two commonly used backbone neural networks on image classification task.

Experiment Setup

- 10 **[00135]** In the experiments, feature extraction and fidelity calculation were performed only on image classification task. Similarly, the features to be compressed in the experiments were extracted by VGGNet-16 and ResNet-50 on a subset of ILSVRC 2012 validation dataset. After feature compression and de-compression, the reconstructed feature maps were sent back to their birth-layer in the corresponding deep
15 learning models to infer the classification results. The compression fidelity were then calculated by comparing the pristine and reconstructed classification results, which is formulated as:

$$Fidelity = 1 - \frac{1}{2NC} \sum_i^N Hamming(Y_p^i, Y_r^i)$$

(Equation 9)

- 20 where Y_p^i is the pristine classification result in form of onehot vector of i -th test sample, Y_r^i is classification result inferred from the corresponding reconstructed feature maps, C is number of classes, N denotes the sample size of test dataset. As to the compression performance, compression rate is applied to reflect the data volume reduction, which is defined as in Equation (4).

25

Comparison of Repack Methods

- [00136]** In the experiments, the intra-channel compression results on classification task were taken as the baseline results. To explore the inter-channel redundancy, three repack methods as described hereinbefore under the section “*Repack*” were tested to
30 assist the video codec. In the experiments, the pre-quantization module 1420 was set as logarithmic mode at 8-bit. Reference software (HM16.12) of HEVC Range extension (RExt) was employed in the video encoder module 1428. To enable inter-channel

redundancy reduction, the video encoder 1428 was set to default Random Access configuration. The compression was performed at five quantization parameter (QP) values, i.e., [0; 12; 22; 32; 42]. Together with the baseline results, compression results with three repack methods on 10 types of feature maps of VGGNet-16 were plotted in FIGs. 20A to 20E. In particular, FIGs. 20A to 20E show plots comparing baseline, naïve channel concatenation, channel concatenation by distance and channel tiling. In FIGs. 20A to 20E, the horizontal axis represents the compression rate, while vertical axis represents fidelity. It means that points closer to the upper left are with higher compression ratio and fidelity. In other words, the closer a curve is to the upper left corner, the more efficient is the corresponding method. From FIGs. 20A to 20E, it can be observed that intra-channel compression (i.e., baseline) and inter-channel compression (i.e., naïve channel concatenation, channel concatenation by distance, and channel tiling) do not have significant performance differences on low layer feature maps (i.e., conv1 to pool3). On the contrary, when the layer becomes higher, inter-channel compression becomes superior to the baseline. It is meaningful since high layer features, such as pool4 to pool5, are the most widely-used features in computer vision tasks. As to the three repack methods, it was observed that channel tiling is notably superior to the channel concatenation methods on high layer features. On low layer features, performance of the three methods varies depending on different feature types, but the performance difference is not performance. As to the two channel concatenation methods, channel concatenation by distance generally achieve better performance at high QP (i.e., QP42).

25 ***Comparison of Pre-Quantization methods***

[00137] To compare the performance maintenance ability of the two methods for pre-quantization module 1420, only quantization and de-quantization were applied on the feature maps of VGGNet-16 and ResNet-50. Information loss under six bit-depth points was evaluated, i.e., [16; 12; 10; 8; 6; 4]-bit. The fidelity of quantized feature by both uniform and logarithmic quantizers at 16-bit and 12-bit all equal to 1:000. The results from 10-bit to 4-bit are listed in Table 4 shown in FIG. 21. In particular, Table 4 shows the fidelity comparison of two pre-quantization methods (uniform and logarithmic) on different feature types and bit depth. As highlighted in Table 4, in most of cases, logarithmic quantization method maintains higher fidelity than the uniform

quantizer on the feature maps. Especially at low bit depth (i.e., 4-bit) and on low layer features of ResNet (i.e., *conv1* and *pool1*), fidelity of logarithmic quantizer can be over 13% higher than the uniform method. In the few cases where logarithmic quantizer is inferior to the uniform, the differences of the two methods are less than 0.4%.

5 Accordingly, the experimental results demonstrate that logarithmic sampling is more suitable for the feature maps quantization in most of the cases. Furthermore, learning-based adaptive quantization may generally achieve better performance than uniform and logarithmic quantization.

[00138] FIGs. 22A and 22B depicts tables (which may be referred to as Tables 5 and
10 6, respectively) listing lossy compression results on VGGNet-16 and ResNet-101 which are the two most widely-used CNNs in computer vision field.

[00139] A lossy compression framework or method for intermediate deep features has been described hereinbefore with reference to FIG. 14A according to various example embodiments. For example, the input is designed to be the deep learning
15 features in single precision floating point (e.g., float32) number. In various example embodiments, the pre-quantization module 1420 may quantize the float32 numbers into lower bit integers (e.g., int8). However, various example embodiments note that with the developing of AI chips and deep learning model quantization techniques, “inference in integer” may be adopted in front-end devices. This means that the intermediate deep
20 features generated in the front-end devices may be integers rather than floating points. To be compatible with the integer inputs, various example embodiments provide a modified compression framework or method 2300 as shown in FIG. 23.

[00140] In particular, FIG. 23 depicts a schematic flow diagram of network-based visual analysis 2300 (e.g., corresponding to the network-based visual analysis as
25 described hereinbefore according to various embodiments) according to various example embodiments of the present invention, and more particular, a method 2304 of visual data transmission for network-based visual analysis (e.g., “encoding process” shown in FIG. 23 and corresponding to the method 300 of visual data transmission for network-based visual analysis as described hereinbefore according to various
30 embodiments) and a method 2308 of network-based visual analysis (e.g., “decoding process” shown in FIG. 23 and corresponding to the method 400 of network-based visual analysis as described hereinbefore according to various embodiments), according to various example embodiments of the present invention. In various example embodiments, the network-based visual analysis 2300 may be the same as that shown

in FIG. 14A or 14B, except that a numerical type determiner 2320/2344 is added to each of the encoding process or module 2304 and the decoding process or module 2308 as shown in FIG. 23.

[00141] In particular, in relation to the encoding process 2304, the numerical type determiner 2320 may be configured to determine whether the plurality of feature maps inputted thereto are in a floating point format (e.g., whether float32 numbers) or in an integer format. If they are in a floating point format, the numerical type determiner 2320 may be configured to direct the plurality of feature maps to the pre-quantization module 1420 to perform quantization thereon as described hereinbefore according to various example embodiments. Otherwise, the numerical type determiner 2320 may be configured to direct the plurality of feature maps to the repack module 1424 to perform repacking thereon as described hereinbefore according to various example embodiments, that is, without subjecting the plurality of feature maps to the pre-quantization module 1420 for quantizing the plurality of feature maps. In various embodiments, the numerical type (e.g., floating point format or integer format) of the plurality of feature maps may be determined based on a numerical type information (e.g., a flag or an identifier) associated with the plurality of feature maps (e.g., associated with the intermediate deep feature).

[00142] In relation to the decoding process 2308, similarly, the number type determiner 2344 may be configured to determine whether the plurality of de-repacked feature maps inputted thereto are based on a plurality of original feature maps 1416 in a floating point format (e.g., whether float32 numbers) or in an integer format. If the plurality of de-repacked feature maps inputted thereto are based on a plurality of original feature maps 1416 in a floating point format, the numerical type determiner 2344 may be configured to direct the plurality of de-repacked feature maps to the pre-dequantization module 1444 to perform pre-dequantization thereon as described hereinbefore according to various example embodiments. Otherwise, the numerical type determiner 2320 may be configured to direct the plurality of de-repacked feature maps to produce the intermediate deep feature without subjecting the plurality of de-repacked feature maps to the pre-dequantization module 1444. In various embodiments, similarly, the numerical type (e.g., floating point format or integer format) of the plurality of original feature maps 1416 may be determined based on a numerical type information (e.g., a flag or an identifier) associated with the plurality of feature maps and transmitted to the visual analysis device at the server end.

[00143] For example, in the encoding phase 2304, the intermediate deep features may be either float feature or integer feature. The numerical type determiner 2320 may be configured to identify the data type (e.g., the numerical type) of the deep features. If the deep features are determined to be float features, they are converted into integer by the pre-quantization module to fit the input requirement of the video encoder 1428 and reduce the data volume. The repack module 1428 may be configured to modify the data shape to fit the input requirement of the video encoder 1428 to maximize the coding efficiency. For the video encoder 1428, existing or conventional video encoders may be applied as desired or as appropriate. By integrating video codecs into the compression framework according to various example embodiments, matured video coding techniques can be borrowed or employed to intermediate feature coding seamlessly. Furthermore, as video encoding/decoding modules (e.g., chips, IP cores, and so on) have already been widely deployed in many cloud based systems, it is economically and technically friendly to upgrade or modify the visual devices and systems to support intermediate deep feature conveyance and analysis with the compression framework according to various example embodiments.

[00144] Accordingly, various example embodiments provide a method which compresses and transmits intermediate deep features, instead of visual signal or ultimately utilized features, in network-based (e.g., cloud-based) visual analysis. The method helps to reduce the computing load at the cloud end, while maintaining the availability of various visual analysis applications, such that better trade-off can be achieved in terms of the computational load, communicational cost and generalization capability. In various example embodiments, a video codec based lossy compression framework and evaluation metrics for intermediate deep feature compression are provided. As discussed hereinbefore, experimental results demonstrated the effectiveness of the network-based visual analysis and the feasibility of the data transmission strategy according to various example embodiments of the present invention.

[00145] While embodiments of the invention have been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come

within the meaning and range of equivalency of the claims are therefore intended to be embraced.