



(51) International Patent Classification:

H04N 19/91 (2014.01) *H04N 19/463* (2014.01)
H04N 19/94 (2014.01)

(21) International Application Number:

PCT/US2023/060460

(22) International Filing Date:

11 January 2023 (11.01.2023)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

17/650,728 11 February 2022 (11.02.2022) US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US];
ATTN: International IP Administration, 5775 Morehouse
Drive, San Diego, California 92121-1714 (US).

(72) Inventors: **SAID, Amir**; 5775 Morehouse Drive, San
Diego, California 92121-1714 (US). **ZHU, Yinhao**; 5775
Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: **EVANS, Matthew J.**; Shumaker & Sieffert, P.A.,
1625 Radio Drive, Suite 100, Woodbury, Minnesota 55125
(US).

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

(84) Designated States (*unless otherwise indicated, for every
kind of regional protection available*): ARIPO (BW, CV,
GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,
TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,
TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,

(54) Title: ENTROPY CODING FOR NEURAL-BASED MEDIA COMPRESSION

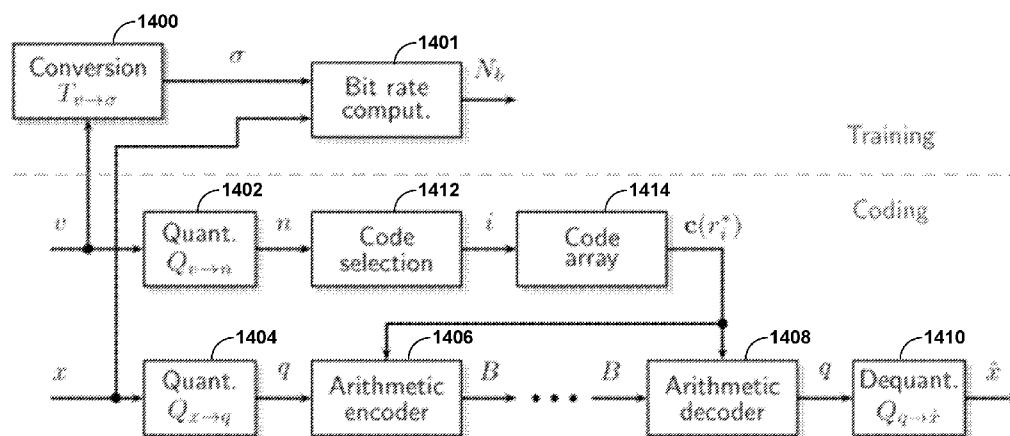


FIG. 14

(57) Abstract: This disclosure describes entropy coding techniques for media data coded using neural-based techniques. A media coder is configured to determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream, determine a code vector based on the probability distribution function parameter, and entropy code the data element using the code vector.



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

Declarations under Rule 4.17:

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

Published:

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

ENTROPY CODING FOR NEURAL-BASED MEDIA COMPRESSION

[0001] This application claims priority to U.S. Patent Application 17/650,728, filed 11 February 2022, the entire content of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] This disclosure relates to media encoding and decoding, including the encoding and decoding of images and video data.

BACKGROUND

[0003] Digital media capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called “smart phones,” video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video coding techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), ITU-T H.265/High Efficiency Video Coding (HEVC), ITU-T H.266/Versatile Video Coding (VVC), and extensions of such standards, as well as proprietary video codecs/formats such as AOMedia Video 1 (AV1) that was developed by the Alliance for Open Media. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video coding techniques.

SUMMARY

[0004] In general, this disclosure describes techniques for media compression, including techniques for video and/or image encoding and decoding. Media (e.g., images and/or video) compression methods based on neural networks can be competitive to current standards and provide several additional advantages. Neural-based coding methods are commonly designed and tested using high-precision floating-

point arithmetic. However, as the technology moves to practical deployment, neural network weights and activation functions are typically quantized and represented with low-precision integers in order to improve speed and power consumption.

[0005] This disclosure addresses the problems that occur when the neural network variables related to entropy coding are quantized. Neural network variables are important to the design of a neural-based video/image compression scheme because such variables define compression efficiency. Furthermore, the very specific properties of the entropy coding variables are not taken into account by general tools for optimizing quantization in neural networks. Testing has shown that the worst quantization effects can occur exactly on some of the most common use cases, and the losses caused by the worst quantization effects cannot be recovered by re-training the neural network.

[0006] This disclosure describes techniques to optimize the definition of the trained entropy coding variables, so that the information that is most important for effective entropy coding is best preserved when represented with low-precision integers. Testing has also shown how the techniques described herein can be used to minimize the amount of memory needed for entropy coding. This disclosure describes a general approach for entropy coding design, as well as specific solutions and implementations for the commonly used Gaussian distributions. The techniques of this disclosure can be generally applied to any neural-based compression techniques, but the examples described below focus on techniques for images and video.

[0007] In one example, a method includes determining a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream, determining a code vector based on the probability distribution function parameter, and entropy coding the data element using the code vector.

[0008] In another example, a device includes a memory, and one or more processors in communication with the memory, the one or more processors configured to determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability

distribution function of the data stream, determine a code vector based on the probability distribution function parameter, and entropy code the data element using the code vector.

[0009] In another example, a computer-readable storage medium is encoded with instructions that, when executed, cause a programmable processor to determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream, determine a code vector based on the probability distribution function parameter, and entropy code the data element using the code vector.

[0010] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, drawings, and claims.

BRIEF DESCRIPTION OF DRAWINGS

[0011] FIG. 1 is a block diagram illustrating an example media encoding and decoding system that may perform the techniques of this disclosure.

[0012] FIG. 2 illustrates differences being conventional media coding and neural-based media coding.

[0013] FIG. 3 illustrates an example of a Gaussian continuous probability distribution function.

[0014] FIG. 4 is a block diagram illustrating an example neural-based image codec.

[0015] FIG. 5 illustrates an example implementation of neural-based entropy coding.

[0016] FIG. 6 illustrates an example implementation of neural-based entropy coding with a quantized probability distribution function parameter.

[0017] FIG. 7 illustrates example measures of redundancy for quantized probability distribution function parameters for different code vectors.

[0018] FIG. 8 illustrates an example of non-uniform assignments of code vectors.

[0019] FIG. 9 illustrates an example of a distribution of Gaussian standard deviations according to bit rates.

[0020] FIG. 10 illustrates an example entropy of quantized random variables.

[0021] FIG. 11 illustrates an example average relative redundancy from using uniform quantization of Gaussian standard deviation.

[0022] FIG. 12 illustrates an example entropy of quantized random variables on a logarithmic scale.

[0023] FIG. 13 illustrates another example entropy of quantized random variables on a logarithmic scale.

[0024] FIG. 14 illustrates a block diagram of another example neural-based codec.

[0025] FIG. 15 illustrates an example of average relative redundancy using uniform quantization of a probability distribution function parameter.

[0026] FIG. 16 illustrates example code for converting a quantized probability distribution function parameter to a code vector.

[0027] FIG. 17 illustrates an example of relative redundancy using a uniform quantization of a probability distribution function parameter.

[0028] FIG. 18 illustrates another example code for converting a quantized probability distribution function parameter to a code vector.

[0029] FIG. 19 is a flowchart illustrating an example method of the disclosure.

DETAILED DESCRIPTION

[0030] This disclosure describes techniques for encoding and decoding media data (e.g., images or videos) using neural-network-based media coding techniques. In particular, this disclosure describes techniques for the entropy coding of media data using optimized probability distribution function (PDF) parameterizations. The techniques of this disclosure include the definition and implementation of a quantized modified PDF parameter that is a non-uniform (e.g., logarithmic) function of a PDF parameter (e.g., in general a standard deviation or any parameter commonly used to define a statistical distribution) used to train the neural networks. By using the PDF parameters of this disclosure, the entropy coding of media data using neural-based coding techniques may be performed with greater efficiency, using less memory, and the performance of the neural-based coding techniques is much less degraded when the neural network weights, activations, and outputs are represented by low-precision integers.

[0031] FIG. 1 is a block diagram illustrating an example media encoding and decoding system 100 that may perform the techniques of this disclosure. In the context of this

disclosure, media may include any digital file to be compressed, including video data and/or images. The example techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data and/or image data. While examples of FIG. 1 will be described with reference to media encoding and decoding, the techniques of this application are equally applicable to the encoding and decoding of any type of data file using neural-based compression techniques.

[0032] As shown in FIG. 1, system 100 includes a source device 102 that provides encoded media data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the media data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0033] In the example of FIG. 1, source device 102 includes media source 104, memory 106, media encoder 200, and output interface 108. Destination device 116 includes input interface 122, media decoder 300, memory 120, and display device 118. In accordance with this disclosure, media encoder 200 of source device 102 and media decoder 300 of destination device 116 may be configured to apply the techniques for entropy coding a neural-based media compression system. Thus, source device 102 represents an example of a media encoding device, while destination device 116 represents an example of a media decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 102 may receive media data from an external media source, such as an external camera. Likewise, destination device 116 may interface with an external display device, rather than include an integrated display device.

[0034] System 100 as shown in FIG. 1 is merely one example. In general, any digital media encoding and/or decoding device may perform techniques for entropy coding a neural-based media compression system. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates

coded media data for transmission to destination device 116. This disclosure refers to a “coding” device as a device that performs coding (encoding and/or decoding) of data. Thus, media encoder 200 and media decoder 300 represent examples of coding devices, in particular, a media encoder and a media decoder, respectively. In some examples, source device 102 and destination device 116 may operate in a substantially symmetrical manner such that each of source device 102 and destination device 116 includes media encoding and decoding components. Hence, system 100 may support one-way or two-way media transmission between source device 102 and destination device 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0035] In general, media source 104 represents a source of media data (i.e., raw, unencoded media data) and provides a sequential series of pictures (also referred to as “frames”) of the media data to media encoder 200, which encodes data for the pictures. Media source 104 of source device 102 may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, media source 104 may generate computer graphics-based data as the source media, or a combination of live media, archived media, and computer-generated media. In each case, media encoder 200 encodes the captured, pre-captured, or computer-generated media data. Media encoder 200 may rearrange the pictures from the received order (sometimes referred to as “display order”) into a coding order for coding. Media encoder 200 may generate a bitstream including encoded media data. Source device 102 may then output the encoded media data via output interface 108 onto computer-readable medium 110 for reception and/or retrieval by, e.g., input interface 122 of destination device 116.

[0036] Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some examples, memories 106, 120 may store raw media data, e.g., raw media from media source 104 and raw, decoded media data from media decoder 300. Additionally or alternatively, memories 106, 120 may store software instructions executable by, e.g., media encoder 200 and media decoder 300, respectively. Although memory 106 and memory 120 are shown separately from media encoder 200 and media decoder 300 in this example, it should be understood that media encoder 200 and media decoder 300 may also include internal memories for

functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded media data, e.g., output from media encoder 200 and input to media decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more buffers, e.g., to store raw, decoded, and/or encoded media data.

[0037] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded media data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded media data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded media data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0038] In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded media data.

[0039] In some examples, source device 102 may output encoded media data to file server 114 or another intermediate storage device that may store the encoded media data generated by source device 102. Destination device 116 may access stored media data from file server 114 via streaming or download.

[0040] File server 114 may be any type of server device capable of storing encoded media data and transmitting that encoded media data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File

Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast Multicast Service (MBMS) or Enhanced MBMS (eMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

[0041] Destination device 116 may access encoded media data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded media data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

[0042] Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 comprise wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded media data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 comprises a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded media data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to media encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to media decoder 300 and/or input interface 122.

[0043] The techniques of this disclosure may be applied to media coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming media transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital media that is encoded onto a data storage medium, decoding of digital media stored on a data storage medium, or other applications.

[0044] Input interface 122 of destination device 116 receives an encoded media bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded media bitstream may include signaling information defined by media encoder 200, which is also used by media decoder 300. Display device 118 displays decoded pictures of the decoded media data to a user. Display device 118 may represent any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0045] Although not shown in FIG. 1, in some examples, media encoder 200 and media decoder 300 may each be integrated with an audio encoder and/or audio decoder, and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and media in a common data stream.

[0046] Media encoder 200 and media decoder 300 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of media encoder 200 and media decoder 300 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including media encoder 200 and/or media decoder 300 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0047] Image and video (e.g., media) compression methods based on neural networks can be competitive to current standards and provide several additional advantages. In

some examples, neural-based coding methods are commonly designed and tested using high-precision floating-point arithmetic. However, use of high-precision floating point arithmetic may impact deployment due to high processing time and power consumption. As the technology moves to practical deployment, neural network weights and activation functions are typically quantized and represented with low-precision integers in order to improve speed and power consumption.

[0048] That is, for practical deployment, neural network variables (e.g., weights and activation functions) may be quantized. However, there may be issues with quantizing of neural network variables.

[0049] This disclosure describes example techniques that may address the problems that occur when the neural network variables related to entropy coding are quantized. Neural network variables related to entropy coding are important to the design of a neural-based video/image compression scheme because such variables define compression efficiency. Furthermore, the very specific properties of the entropy coding variables are not taken into account by general tools for optimizing quantization in neural networks. Testing has shown that, in fact, the worst quantization effects can occur exactly on some of the most common use cases, and the losses caused by the worst quantization effects cannot be recovered by re-training the neural network.

[0050] This disclosure describes techniques that further optimize the definition of the entropy coding variables so that the information that is most important for effective entropy coding is best preserved. Testing has also shown how the techniques described herein can be used to minimize the amount of memory needed for entropy coding. This disclosure describes a general approach for entropy coding design, as well as specific solutions and implementations for the commonly used Gaussian distributions.

[0051] As will be explained in more detail below, media encoder 200 and media decoder 300 may be configured to determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is derived from a logarithmic function of a standard deviation of a probability distribution function of the data stream, determine a code vector based on the probability distribution function parameter, and entropy code the data element using the code vector.

[0052] Entropy coding is a fundamental part of media compression systems. The entropy coding process is responsible for optimizing the conversions between media information and the compressed data bitstreams, aiming to obtain the most compact representation possible. Unlike other elements of media compression, entropy coding is a lossless process, i.e., it fully preserves information.

[0053] Several techniques were developed for implementing efficient entropy coding in image and video compression standards. More recently it has been shown that new compression methods based on deep learning and neural networks are approaching the performance of conventional methods, while offering several other practical advantages.

[0054] Because conventional and neural-based codecs use very distinct forms of compression, conventional and neural-based codecs employ entropy coding in significantly different ways. A summary of the main differences is shown in FIG. 2, where it can be observed that, even if the two approaches use the same coding method, like arithmetic coding, all the other aspects are different. This means that it is beneficial to develop new techniques to optimize the performance in neural-based codecs.

[0055] As shown in FIG. 2, the design phase of both conventional and neural-based coding techniques include data organization process 200 and statistical data modeling process 202. For conventional media coding techniques, such as H.265, data organization process 200 includes the definition of coding elements (e.g., syntax elements and coding modes). For neural-based coding processes, data organization process 200 includes the training and development of image/video (e.g., media) autoencoder networks. For statistical data modeling process 202, conventional coding techniques include the coding of context definitions, e.g., for an entropy coding process, such as context-adaptive binary arithmetic coding. For neural-based coding processes, statistical modeling process 202 includes the development and training of a hyperprior neural network.

[0056] Both conventional and neural-based coding techniques include an implementation phase that includes probability determination process 204 and conversion of information into bits process 206. Both the probability determination process 204 and conversion of information into bits process 206 are based on the data to be coded. For conventional coding, probability determination process 204 (e.g., a bit-rate assignment process) includes adaptive context probability estimation. For neural-based coding, probability determination process 204 includes the determination of

probability distribution parameters, as will be described in more detail below. Both the conventional and neural-based coding techniques use an arithmetic and/or entropy coder (e.g., or arithmetic or entropy encoder or decoder) for the conversion of information into bits process 206. The arithmetic coder may be a Huffman, range or asymmetric numeral system (ANS) coding.

[0057] The data stream management phase is the management of compressed data 208. In some example conventional coding systems, the compressed data is in a single bitstream. Some example neural-based coding systems use multiple streams for concurrent coding.

[0058] A main entropy coding principle, derived from basic information theory principles, is that the optimal number of bits N_b to be assigned for coding a data symbol is:

$$N_b = -\log_2(\hat{p}), \quad (1)$$

where $\hat{p}$ is the estimated probability of the value of the symbol to be encoded, with knowledge of $\hat{p}$ shared by the encoder and decoder (e.g., media encoder 200 and media decoder 300). This optimal number for the average number of bits is a fractional number, but it can be effectively achieved in practice using arithmetic coding.

[0059] Symbol probabilities can vary widely and frequently, depending on the specific parts of the media content. Thus, in conventional media compression, one of the most important tasks is the efficient estimation of those data symbol probabilities, and how to accurately update the data symbol probabilities as they change during coding. Since this probability estimation is done sequentially, it can be very difficult to parallelize entropy coding on conventional video coding methods.

[0060] Compression methods proposed for neural-based codecs are designed to work with the assumption that the data to be compressed has a certain type of probability density function (PDF) and only the parameters defining this function can vary. As one example, the PDF can be chosen to have all variables with independent zero-mean Gaussian distributions, each with a different standard deviation σ . This may not simply be an empiric assumption since the networks “learn” to make it real. The PDF is typically, but not always, Gaussian, as shown in the example of FIG. 3. FIG. 3 illustrates an example of a Gaussian continuous PDF 300 commonly used in neural-based codecs, the quantized values, and respective probabilities ($p_0, p_1, \dots, p_8$). Multiple values corresponding to rare underflow and overflow cases are typically coded

sub-optimally with methods that do not require probability estimates.

[0061] In general, σ is a scaling factor for a “template” distribution. For example, the Gaussian or Laplace with zero mean and unit variance are templates, and there are scaling parameters to “stretch” or “shrink” the template. Commonly the standard deviation is used because of its convenience, but it could be any factor. This disclosure defines techniques that change from a parameter or scale factor that is convenient for humans (e.g., traditionally used by statisticians) to one that is optimized for entropy coding using low-precision parameter representations.

[0062] In some examples, a separate neural network (e.g., separate from the neural network that performs the media compression) is trained to determine the parameters of the distributions. Details of this training will be described below. Before entropy coding, those variables are discretized with a fixed quantization scheme, and the probabilities of the discrete values ($p_0, p_1, \dots, p_8$, in the example of FIG. 3), are derived from the assumed distribution, and later used for determining the number of symbol bits according to eq. (1).

[0063] Unlike conventional coding methods where it is typically necessary to estimate the probabilities for all possible values of each data symbol, neural-based methods typically rely solely on the parameters of the PDF (e.g., standard deviation σ), which are computed in parallel using neural networks, making concurrent entropy coding much easier to implement, without compression loss. Since entropy coding is completely defined by values of those PDF parameters, compression performance can be severely degraded when the PDF parameters are represented with low integer precision. This disclosure describes new techniques that minimize those compression losses.

[0064] One important class of methods proposed for neural-based image and video compression adapts the variational autoencoder architecture, where the latent variables are quantized and entropy coded using fixed learned distributions. In some examples, this architecture has been extended by adding a smaller neural network, called *hyperprior*, that uses the variables generated from the autoencoder to define the probability distributions to be used for entropy coding those autoencoder variables. This hyperprior network also uses quantization and entropy coding, but with pre-defined (e.g., learned) PDF parameters. FIG. 4 is a diagram of a neural image codec using variational autoencoder and hyperprior networks for entropy coding.

[0065] The process shown in FIG. 4, where the neural networks for image analysis and synthesis implement the variational autoencoder, and entropy coding of its variables is defined by the hyperprior decoder network (used at both sender and receiver). Since *encoder* and *decoder* (e.g., media encoder 200 and media decoder 300) have been used to denote autoencoder parts, in the neural network literature it is common to refer to the systems traditionally called by those names as *sender* and *receiver*, respectively.

[0066] In FIG. 4, media encoder 200 may include image analysis neural network 402, hyperprior encoder neural network 404, quantization process 406, entropy encoding process 408, quantization process 416, and entropy encoding process 418. Media encoder 200 may also include entropy decoding process 412, hyperprior decoder neural network 424, entropy decoding process 422, and image synthesis neural network 426 (which may be similar to a reconstruction loop in a hybrid video coder). Though not shown, media decoder 300 may include similar structures to entropy decoding process 412, hyperprior decoder neural network 424, entropy decoding process 422, and image synthesis neural network 426.

[0067] Image analysis neural network 402 is a neural network configured to encode and/or compress input image 400. The compressed data stream created by image analysis neural network 402 is then processed by hyperprior encoder neural network 404. The output of hyperprior encoder neural network 404 is then quantized by quantization process 406 and entropy coded by entropy encoding process 508 to create bitstream 1 410. Hyperprior encoder neural network 404 is part of the process configured to determine parameters (e.g., a standard deviation) of a PDF for the data stream created by image analysis neural network 402. The parameters of the PDF may then be used to entropy encode the data stream from image analysis neural network 402.

[0068] Entropy encoding process 408 itself uses PDF parameters (e.g., learned parameter 414) that are learned from a plurality of images to be encoded. Entropy decoding process 412 uses the same PDF parameters to entropy decode bitstream 1 410. This entropy decoded bitstream 412 is then processed by hyperprior decoder neural network 424 in order to generate the PDF parameters for the specific image (e.g., input image 400) being decoded.

[0069] The PDF parameters generated by hyperprior decoder neural network 404 are then used by entropy encoding process 418 to encode the data stream produced by image analysis neural network 402 after quantization by quantization process 416.

Entropy encoding process 418 create bitstream 2 420. Bitstream 2 420 is then entropy decoded by entropy decoding process 422 using the same PDF parameters from hyperprior decoder neural network 424. After entropy decoding, image synthesis neural network 426 processes the decoded data to produce output image 428.

[0070] FIG. 5 provides a more detailed diagram of the coding process of a single data element denoted by x . In particular, FIG. 5 illustrates an implementation of entropy coding using arithmetic coding, where x is the data to be coded, and vector of PDF parameters σ , known by the sender and receiver (e.g., media encoder 200 and media decoder 3), contains the parameters that define the PDF of x . More specifically, the vector σ is standard deviation (e.g., of a normal gaussian distribution), and more generally is a parameter determined by the hyperprior network to determine the pdf to use for entropy coding.

[0071] In the context of this disclosure, data elements (x) may include inputs to a neural network (e.g., inputs to image synthesis neural network 426 of FIG. 4), outputs of a neural network (e.g., outputs of image analysis neural network 402 of FIG. 4), syntax elements, or other coded data of an encoded media bitstream. Some example types of data that may be encoded to generate data elements x (or more generally, data elements to be entropy coded using the techniques of this disclosure) in neural-based image/video coding may include RGB or YUV pixel values in images or video frames, residuals of pixel values (pixel values minus prediction from motion) in motion compensated video, motion vectors for block-based motion compensation, and/or dense optical flow data for warping-based motion compensation. In other examples, data elements x may include data representing weights, activation functions, or other data used by a neural-based media compression technique.

[0072] Data is not encoded during training phase, but rather bit rate computation process 500 uses data element x and PDF parameter σ to estimate the number of bits N_b used by the arithmetic coding process. The vector σ represents the parameter defining the PDF of data elements x , which is known by the sender and receiver (e.g., media encoder 200 and media decoder 300). The PDF parameter σ may be computed by a hyperprior decoder network, or may be read from a table of predefined parameters that are learned. As one example, code vector computation process 502 may compute a code vector $c(\sigma)$ for a particular data element x from its corresponding PDF parameter σ .

[0073] Quantization process ($Q_{x \rightarrow q}$) 504 quantizes data element x to create quantized data element q . Arithmetic encoder 506 then uses code vector $\mathbf{c}(\sigma)$ to arithmetically code quantized data element q to create bitstream B . Arithmetic decoder 508 recovers quantized data element q by performing arithmetic decoding using the same code vector $\mathbf{c}(\sigma)$. Dequantization process ($Q_{q \rightarrow \hat{x}}$) 510 performs dequantization quantized data element q to recover decoded data element $\hat{x}$. Decoded data element $\hat{x}$ may then be decoded by a neural-based decoding technique to reconstruct the original media (e.g., a picture or video).

[0074] Note that the techniques of this disclosure are applicable to all compression methods containing neural networks that generate parameters of probability distributions used for entropy coding, and also applicable to tables of entropy coding parameters stored with low-precision integers. Even though the next sections frequently use examples where σ represents the standard deviation of Gaussian PDFs, the general techniques of this disclosure are applicable for all cases where a PDF is defined by a parameter σ .

[0075] A PDF may be defined by more than one parameter, but not all parameters may be directly used for entropy coding. For example, a PDF may be defined by standard deviation σ and mean value μ , but only the standard deviation determines entropy coding properties. Since having one parameter is the most common in practice, it is the assumption used in the next sections, and the techniques described below may be easily generalized.

[0076] As shown in FIG. 5, the block called code vector computation process 502 defines how the PDF is used to create a code vector $\mathbf{c}(\sigma)$, which is the information needed by the arithmetic coding process to create bit stream B . For example, the code vector can be a vector with the *cumulative distribution function* (CDF) corresponding to the *probability mass function* (PMF) defined by symbols after quantization ($p_0, p_1, \dots, p_8$, in FIG. 3). In another example, if a technique called *binarization* is used, then code vector $\mathbf{c}(\sigma)$ contains the probabilities of binary decisions to code the same data. In practice, the CDF and probabilities are scaled, so that elements of vector $\mathbf{c}$ are represented as integers.

[0077] The system shown in FIG. 5 represents how entropy coding is used when a neural-based codec is being first designed, trained, and tested, using very precise floating-point arithmetic and powerful processors. However, in practical

implementations, the neural network values are typically represented with low precision integers to enable faster computations and small power usage. This is shown in the modified system of FIG. 6, where an additional quantization block is added to indicate that parameter σ is converted to low precision version $\hat{\sigma}$, represented, for example, as a single 8-bit integer (byte).

[0078] FIG. 6 illustrates a practical implementation of the system of FIG. 5, where PDF parameter σ is quantized and represented by low precision integer $\hat{\sigma}$, and the vectors needed for arithmetic coding are pre-computed and stored in an array with C elements. Like the example of FIG. 5, data is not encoded during training phase, but rather bit rate computation process 600 uses data element x and PDF parameter σ to estimate the number of bits N_b used by the arithmetic coding process. In the example of FIG. 6, quantization process ($Q_{\sigma \rightarrow \hat{\sigma}}$) 602 quantizes the PDF parameter σ to create quantized PDF parameter $\hat{\sigma}$. Code array process 603 then generates code vector $c(\hat{\sigma})$ from quantized PDF parameter $\hat{\sigma}$. That is, quantized PDF parameter $\hat{\sigma}$ is used to determine one of the C elements (e.g., code vectors) available to be determined by code array process 603.

[0079] Quantization process ($Q_{x \rightarrow q}$) 604 quantizes data element x to create quantized data element q . Arithmetic encoder 606 then uses code vector $c(\hat{\sigma})$ to arithmetically code quantized data element q to create bitstream B . Arithmetic decoder 608 recovers quantized data element q by performing arithmetic decoding using the same code vector $c(\hat{\sigma})$. Dequantization process ($Q_{q \rightarrow \hat{x}}$) 610 performs dequantization quantized data element q to recover decoded data element $\hat{x}$. Decoded data element $\hat{x}$ may then be decoded by a neural-based decoding technique to reconstruct the original media (e.g., a picture or video).

[0080] Note that in FIG. 6, the quantization of x is different than the quantization of PDF parameter σ . Floating point parameters may work well for very complex neural networks, but are not very feasible for widespread use for smaller devices (e.g., mobile devices). Power consumption in such mobile devices would be too high. As such, the PDF parameter σ is quantized to an integer for more practical application. The PDF parameter $\hat{\sigma}$ is a 1 byte integer, in one example (e.g., 256 possible values). Such a quantization leads to more power efficiency.

[0081] Another practical modification is motivated by the fact that it is commonly computationally expensive to re-compute code vector $\mathbf{c}(\sigma)$ for each value of x . For example, if x has a Gaussian distribution with standard deviation σ (e.g., a typical case), then the PMF of quantized values is defined by:

$$p_\sigma[n] = \frac{1}{2} \left[\operatorname{erf} \left(\frac{n + 1/2}{\sqrt{2}\sigma} \right) - \operatorname{erf} \left(\frac{n - 1/2}{\sqrt{2}\sigma} \right) \right], \quad (2)$$

where the *error function* is:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \quad (3)$$

[0082] The *entropy* of this PMF, in bits, is:

$$H(\sigma) = - \sum_{n=-\infty}^{\infty} p_\sigma[n] \log_2(p_\sigma[n]). \quad (4)$$

[0083] Note that besides computing probabilities, additional tasks may be performed to create vector $\mathbf{c}$ with the correct integer cumulative sums or binary probabilities, adding to the computational costs. Those computations can be avoided in practical implementations by pre-computing code vectors $\mathbf{c}(\hat{\sigma})$ for different quantized values of parameter $\hat{\sigma}$, and storing those vectors in an array (e.g., in memory). As shown in the diagram of FIG. 6, sender and receiver (e.g., media encoder 200 and media decoder 300) have the same array of code vectors, which are selected according to parameter $\hat{\sigma}$, and used for arithmetic encoding and decoding.

[0084] For example, given the following definition:

$$\hat{\sigma} = Q_\sigma(\sigma) = \lfloor \sigma \rfloor, \quad (5)$$

where $\lfloor \sigma \rfloor$ (floor function) is the greatest integer less than or equal to σ , then x can be entropy coded with pre-computed code vectors $\mathbf{c}(\hat{\sigma}) = \mathbf{c}(\lfloor \sigma \rfloor + 1/2)$.

[0085] The loss incurred by having a limited number of code vectors can be measured using the *code redundancy* function, which is the average number of additional bits needed to encode x using a code vector $\mathbf{c}(r)$ instead of the optimal code vector $\mathbf{c}(\sigma)$.

[0086] The *code redundancy* function is defined by the Kullback–Leibler divergence measure, as follows:

$$R(\sigma|r) = \sum_{n=-\infty}^{\infty} p_\sigma[n] \log_2 \left(\frac{p_\sigma[n]}{p_r[n]} \right). \quad (6)$$

[0087] Entropy coding is not normally designed to minimize an average redundancy, because media codecs operate on a wide range of bit ratios. Commonly, the average is not known *a priori*. Instead, entropy coding is designed by limiting the *relative redundancy*, as follows:

$$L(\sigma|r) = \frac{R(\sigma|r)}{H(\sigma)}. \quad (7)$$

[0088] Designing entropy coding methods that have strictly limited relative redundancy guarantees that all averages are also limited. In other words, with this approach the average compression losses may not be minimal, but they are guaranteed to be small when relative losses are always small.

[0089] FIG. 7 shows examples of those redundancy measures for Gaussian distributions when the code vectors are computed using the following:

$$r = \lfloor \sigma \rfloor + \frac{1}{2}. \quad (8)$$

[0090] FIG. 7 shows examples of absolute measures (plots 700, right Y axis) and relative measures (plots 702, left Y axis) of redundancy related to the use of quantized PDF parameters and an integer number of code vectors. This loss is coding efficiency. Essentially more bits than necessary are used. Quantizing the sigma value results in uncertainty as to the actual value of the floating point sigma.

[0091] It may be assumed that code vectors $\mathbf{c}(1.5)$, $\mathbf{c}(2.5)$, and $\mathbf{c}(3.5)$ are used to entropy code values with PDF parameter σ in intervals $[1, 2)$, $[2, 3)$, and $[3, 4)$, respectively. It can be observed that the redundancy is zero when $\sigma = r$, and otherwise positive. Note that, in the example of FIG. 7, the maximum redundancy in interval $[1, 2)$ is larger than in the other intervals.

[0092] Testing has shown that the maximum redundancy in an interval depends on the width of the interval, and on the values within the interval. This means that better results can be obtained with non-uniform assignment of code vectors.

[0093] FIG. 8 shows an example of non-uniform assignment of code vectors $\mathbf{c}(\sigma)$ designed to obtain same maximum relative redundancy in all intervals. The relative redundancy graphs in FIG. 8 may be obtained if code vectors $\mathbf{c}(1.24)$, $\mathbf{c}(1.93)$, and $\mathbf{c}(3.14)$ are used to entropy code values with PDF parameter σ respectively in intervals $[1, 1.52)$, $[1.52, 2.41)$, and $[2.41, 4)$. It can be observed that with those values the maximum relative redundancy is the same in the three intervals.

[0094] Note that the techniques described above with reference to FIG. 8 are related to code vector assignments, and not the quantization of PDF parameter σ . For instance, if σ is directly quantized to integer values, then the fractional interval thresholds shown in FIG. 8 cannot be applied.

[0095] Given an interval $[\sigma_a, \sigma_b)$ defining the ranges of σ values to be entropy coded with the same code vector $\mathbf{c}(r^*(\sigma_a, \sigma_b))$, a media coder (e.g., media encoder 200 and media decoder 300) may minimize the maximum relative redundancy within this interval if the following equation is used:

$$r^*(\sigma_a, \sigma_b) = \{r: L(\sigma_a|r) = L(\sigma_b|r)\}. \quad (9)$$

[0096] With the above design criterion for code vectors, it can be observed in FIG. 8 that the redundancy within interval $[\sigma_a, \sigma_b)$ has a nearly parabolic shape, with zero value when $\sigma = r^*(\sigma_a, \sigma_b)$. Under those conditions, and assuming a uniform conditional distribution of σ within the interval, an average redundancy may be well approximated by:

$$\bar{L}(\sigma) = \frac{L(\sigma_a|r^*(\sigma_a, \sigma_b))}{3} = \frac{L(\sigma_b|r^*(\sigma_a, \sigma_b))}{3} \approx E_{\sigma \in [\sigma_a, \sigma_b]} \{L(\sigma|r^*(\sigma_a, \sigma_b))\}. \quad (10)$$

[0097] This average is shown as dashed line 800 in the example of FIG. 8. FIG. 8 shows that when several intervals have the same maximum redundancies according to eq. (9), then those intervals also have approximately the same average redundancy.

[0098] One fact to be considered for identifying the problems with the implementation shown in FIG. 6 is that codec designers may choose a type of PDF and the parameters to be used for its definition. However, it is the practical application of the design that defines the relative frequencies of parameter values, since those are defined by network training. For example, FIG. 9 shows a plot 900 of the relative frequency of Gaussian standard deviation values from a neural-based image codec. As shown in FIG. 9, for low bit rates (the most used in practice), σ values are peaked at the smallest allowed value, $\sigma = 0.1$, and nearly all are smaller than $\sigma = 10$. On the other hand, for larger bit rates, larger σ values are more common. This disclosure assumes that, to cover a wide range of reproduction quality, standard deviations are in the range $[0.1, 1,000]$.

[0099] PDF parameter σ defines the bit rates used by the codec, which are measured with the entropy function defined in eq. (4). FIG. 10 shows a plot 1000 that illustrates how the entropy of quantized values varies with parameter σ in the range of standard deviations that are desired to be supported. More specifically, FIG. 10 shows entropy (in

bits) of quantized random variables with Gaussian distribution as a function of the standard deviation σ .

[0100] If a media coder (e.g., media encoder 200 or media decoder 300) is configured to quantize σ in the range [0.1, 1,000] to 256 values (fitting a single byte), and uses eq. (9) to create one code vector for each of the 256 intervals defined by $\hat{\sigma}$, the media coder would obtain the average values of relative redundancy that are shown in FIG. 11. FIG. 11 shows a plot 1100 illustrating the average relative redundancy (loss in compression) incurred by using uniform quantization of Gaussian standard deviation σ , and $C = 256$ code vectors.

[0101] The following problems can be observed for FIGS. 10 and 11:

1. The entropy function has very high derivatives at small values of σ (the most used range), complicating the learning convergence during training.
2. Because the entropy is highly non-linear near the origin, it is difficult to be reproduced with quantized neural networks.
3. Important differences for small σ values cannot be separated when $\hat{\sigma}$ has low precision, resulting in the very high relative redundancy shown near the origin in FIG. 11.
4. The range of σ corresponds to values that are needed in practice, but as shown in FIG. 9, most of that range is typically rarely used. In the example shown, the range of small values of values of σ is most used, and it is also where the quantization problems are worst.

[0102] In short, using of the standard deviation as the value for defining entropy coding can work reasonably well with high-precision floating-point implementations, but may not be beneficial for the quantized neural networks used in practical implementations.

[0103] Similar problems occur for other types of probability distributions whenever codec designers choose a PDF parameter because it has a familiar and intuitive meaning. High precision neural networks can “learn” how to optimally use those parameters via extensive training, but quantized networks may not. Using non-uniform quantization (e.g., as shown in the example of FIG. 8) may not directly solve those problems, since in the practical implementation, the assignment of code vectors is made using quantized parameter $\hat{\sigma}$.

[0104] The practical problems presented above are caused by how PDF parameter σ shapes the entropy function, and by its conversion to a low-precision integer

representation (indicated in FIG. 6 by the block with quantization Q_σ). It may be difficult to identify the origins of those problems from a direct analysis of how eq. (4) defines the entropy of quantized values, since it is an infinite sum of terms without a clear intuitive interpretation. More insight may be obtained by considering simpler approximations. For Gaussian distributions and large values of σ the entropy can be approximated by the following:

$$H(\sigma) \approx \log_2(\sigma + 0.05) + \frac{1 + \ln(2\pi)}{2 \ln(2)}, \quad (11)$$

and for small values of values of σ by:

$$H(\sigma) \approx \frac{(4\sigma + 0.35)}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{8\sigma^2}\right). \quad (12)$$

[0105] Error! Reference source not found. 12 shows the entropy graph 1200 of FIG. 10 when a logarithmic scale is used for parameter σ , together with the approximations defined above. In both, quick asymptotic convergence can be observed, as well very good matches within the proper range of σ , and a middle range where both approximations are nearly the same.

[0106] From FIG. 12, it is easier to recognize that the logarithm of σ provides a more direct relationship to the entropy, and thus can be a better parameter to define the PDF. One remaining aspect to be considered is that, as shown in FIG. 9, small σ values are expected to be the most common, and such small values for σ correspond to very small entropies.

[0107] Using logarithmic scales for both entropy and standard deviation, as shown in FIG. 13, shows that the way the entropy function varies at low values of σ may also be beneficial to consider. Since it may be that case that a media encoder 200 or media decoder 300 may encounter a very large number of values with very low entropy, even very small absolute errors can produce large relative redundancies and large compression losses. FIG. 13 shows a plot 1300 that illustrates entropy and its approximations from eqs. (11) and (12), graphed with logarithmic scales.

[0108] Even though quantization errors are inevitable, the techniques of this disclosure are based on the fact that it is possible to reduce the compression loss by replacing the variable to be quantized, considering how the quantization errors affect the relative redundancy.

[0109] One example technique is shown in FIG. 14, where a new variable called v is defined for parameterizing the PDF. In general PDF parameter v may be determined using a non-linear function (e.g., logarithmic) of PDF parameter σ . A conversion function between σ and v is defined such that:

$$\sigma = T_{v \rightarrow \sigma}(v). \quad (13)$$

[0110] FIG. 14 shows a modification of the practical codec of FIG. 6 to enable new design variables optimized for entropy coding, and tables for more efficient assignment of code vectors. As indicated in FIG. 14, this function allows the codec designers to continue using PDF parameter σ in training, and it is assumed that this function is differentiable almost everywhere, to preserve automatic gradient back propagation.

[0111] To simplify the presentation, the following conventions are adopted:

$$\begin{aligned} \sigma &\in [\sigma_{\min}, \sigma_{\max}) \subset \mathbb{R}, \\ v &\in [0, 1) \subset \mathbb{R}, \\ T_{v \rightarrow \sigma}(0) &= \sigma_{\min}, \\ T_{v \rightarrow \sigma}(1) &= \sigma_{\max}, \\ Q_{v \rightarrow n}(v) &= [N v] \in \{0, 1, \dots, N-1\}, \\ v_0 < v_1 &\Rightarrow T_{v \rightarrow \sigma}(v_0) < T_{v \rightarrow \sigma}(v_1). \end{aligned} \quad (14)$$

[0112] Equation (14) specifies that σ ranges from a minimum value ($\sigma_{\min}$) to a maximum value ($\sigma_{\max}$). New parameter v ranges from 0 to 1. The conversion function $T_{v \rightarrow \sigma}$ at v value of 0 is equal to $\sigma_{\min}$. The conversion function $T_{v \rightarrow \sigma}$ at v value of 1 is equal to $\sigma_{\max}$. The quantization of new parameter v (e.g., $Q_{v \rightarrow n}(v)$) produces integers ranging from 0 to $N-1$. Further it is defined that if $v_0 < v_1$ then $T_{v \rightarrow \sigma}(v_0) < T_{v \rightarrow \sigma}(v_1)$.

[0113] In FIG. 14 data is not encoded during training phase. Because the coding phase is configured to use new PDF parameter v , the training phase includes conversion ($T_{v \rightarrow \sigma}$) process 1400 to convert PDF parameter v back to PDF parameter σ (e.g., the standard deviation of the PDF). Bit rate computation process 1401 uses data element x and PDF parameter v to estimate the number of bits N_b used by the arithmetic coding process.

[0114] In the example of FIG. 14, quantization process ($Q_{v \rightarrow n}$) 1402 quantizes the PDF parameter v to create quantized PDF parameter n . Code selection process 1412 then converts the quantized PDF parameter n to a code vector index i . Code vector index i may be an input to code array process 1414. That is code vector index i may be an index to a lookup table of code vectors. Code array process 1414 then determines code vector $c(r_i^*)$ from code vector index i . Code vector $c(r_i^*)$ is used to arithmetically code

data element x .

[0115] Quantization process ($Q_{x \rightarrow q}$) 1404 quantizes data element x to create quantized data element q . Arithmetic encoder 1406 then uses code vector $c(r^*_i)$ to arithmetically code quantized data element q to create bitstream B . Arithmetic decoder 1408 recovers quantized data element q by performing arithmetic decoding using the same code vector $c(r^*_i)$. Dequantization process ($Q_{q \rightarrow \hat{x}}$) 1410 performs dequantization quantized data element q to recover decoded data element $\hat{x}$. Decoded data element $\hat{x}$ may then be decoded by a neural-based decoding technique to reconstruct the original media (e.g., a picture or video).

[0116] In general, in one example of the disclosure, media encoder 200 and media decoder 300 may be configured to determine a probability distribution function parameter (e.g., v) for a data element (x) of a data stream coded by a neural-based media compression technique. The probability distribution function parameter (v) may be determined from a logarithmic function (e.g., natural logarithm ($\ln$)) of a standard deviation (σ) of a probability distribution function of the data stream. Media encoder 200 and media decoder 300 may be further configured to determine a code vector ($c(r^*_i)$) based on the probability distribution function parameter (v), and may entropy code the data element (x) using the code vector ($c(r^*_i)$).

[0117] As shown in FIG. 14, in one example, media encoder 200 and media decoder 300 may be configured to quantize the probability distribution function parameter v prior to determining the code vector. In one example, to determine the code vector media encoder 200 and media decoder 300 may be configured to determine a code vector index (i) based on the probability distribution function parameter v , and determine the code vector from a table of code vectors based on the code vector index. More information on tables of code vectors will be described below.

[0118] From the encoder side, media encoder 200 may be configured to generate the data element x using the neural-based compression technique, and may quantize the data element to create a quantized data element. In this example, entropy coding the data element using the code vector comprises entropy encoding the quantized data element using the code vector. Further in this example, e.g., as shown in FIG 4, media encoder 200 may be configured to process an image or video picture using an image analysis neural network to generate the data element.

[0119] From the decoder side, media decoder 300 may decode an encoded data element using the code vector to create a quantized data element. Media decoder 300 may further dequantize the quantized data element to create a reconstructed data element. Media decoder 300 may then process the reconstructed data element using an image synthesis neural network to reconstruct an image or video picture.

[0120] The next aspect to be considered is that the performance of entropy coding is measured by the relative redundancy measures defined in eqs. (7) and (10). Below, examples of the practical application of this approach are presented.

[0121] **Logarithmic Conversion**

[0122] Most commonly, neural-based codecs are configured to operate using the assumption $x \sim \mathcal{N}(0, \sigma)$, i.e., the variables to be coded have zero-mean Gaussian (or normal) distribution. While some example hyperprior network use some learned PDFs, more recent work show that there is improvement when the learned PDFs are replaced with Gaussian PDFs and learned parameters.

[0123] Using this fact, and the observation in FIG. 12 and FIG. 13 that the logarithm of the standard deviation provides a more precise parameterization of the PDF, a variable conversion function may be defined using:

$$\sigma = T_{v \rightarrow \sigma} = \exp(v \ln(\sigma_{\max}) + (1 - v) \ln(\sigma_{\min})), \quad (15)$$

and its inverse as:

$$v = T_{\sigma \rightarrow v}(\sigma) = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}. \quad (16)$$

[0124] The function $\ln()$ is a natural logarithm. When this new parameter v is quantized using $N = 256$, and the corresponding $C = 256$ code vectors are optimized according to eq. (9), the average relative redundancies shown in FIG. 15 are obtained. FIG. 15 shows plot 1500 which illustrates the average relative redundancy (loss in compression) incurred by using uniform quantization of PDF parameter v with $N = 256$, and $C = 256$ code vectors. FIG. 15 shows that PDF parameter v is a more suitable parameter for defining Gaussian PDFs since the relative redundancy is always below 1%.

[0125] The conversion equations (15) and (16) shown above only use the exponential and logarithm functions, which are continuous and with derivatives that are easy to compute during design and training.

[0126] **Tables for Selecting Code Vectors**

[0127] One additional aspect to be considered is that, since code vectors are constantly accessed during entropy coding, to maintain high data throughputs it may be preferable to store code vectors in expensive fast memory. Thus, reducing the number of code vectors enables savings in memory requirements and costs.

[0128] A savings in memory requirements and costs can be achieved using the code selection tables, which are shown in FIG. 16 after quantization of PDF parameter v . Those are simple tables with N elements, to enable non-uniform assignment of code vectors, similarly to the example of FIG. 8, but instead of using floating-point values defining transitions based on PDF parameter σ , the tables are applied directly to integer values n , obtained after quantization of PDF parameter v .

[0129] The advantage is that this process is an integer-to-integer conversion that can be done very efficiently with a simple table-look-up implementation. FIG. 16 shows an example of such a table 1600 defining the number of code vectors that should be used for entropy coding, for each of the 256 possible values of n . The total table memory is 256 bytes in the example of FIG. 16. Media encoder 200 and media decoder 300 may use table 1600 in FIG. 16 is used to convert PDF parameter v , quantized by value n with $N = 256$, to code vector index i . Table 16 was designed to better ensure maximum redundancy smaller than 1%, as shown in FIG. 17, thus enabling the reduction of the number of code vectors from 256 to 43. FIG. 17 shows a plot 1700 illustrating the resulting relative redundancy according to standard deviation.

[0130] Another way to limit memory usage is to reduce the range of standard deviation values. For example, a neural-based may be implemented using the following PDF parameters:

$$\sigma_{\min} = 0.112, \sigma_{\max} = 334.5 \Rightarrow \ln(\sigma_{\max}/\sigma_{\min}) = 8, \quad (17)$$

which corresponds to following conversion equation:

$$n = \max [0, \min [255, [32 \ln(\sigma)] + 70]], \quad (18)$$

where multiplication by 32 can be implemented via bit shifts.

[0131] FIG. 18 shows a code vector selection table 1800 designed for the above transformation, reducing the number of code vectors to 25.

[0132] The techniques of this disclosure may be implemented and tested using arithmetic coding and integrated into neural-based codecs running on a mobile device. In one example, the arithmetic coding techniques may be implemented using 32-bit operations, and the probability values may scaled to 16 bits.

[0133] Table 1 shows some experimental results used to confirm the performance of the techniques of this disclosure. The hyperprior data was obtained using floating-point operations, so that only the quantization of entropy coding parameters affected the results.

[0134] As shown by the first row of results in Table 1, when the quantized PDF parameter σ is used to parameterize the Gaussian PDF, about 0.5 Mbytes of memory is needed for storing 256 code vectors, and the obtained redundancy is what is expected from FIG. 11, approaching 1000%. Better results are possible, but would require much more memory for code vectors.

[0135] The following rows in Table 1 show that much better results are obtained when σ is replaced by quantized PDF parameter v (e.g., a parameter obtained using a logarithmic function of σ), as defined by eq. (16). Using $N = C = 256$ practically eliminates the redundancy, but the amount of memory for the code vectors is relatively high.

[0136] The results show that using smaller values of $N = C$ helps to reduce the amount of memory needed, but the relative redundancy increases non-uniformly. On other hand, using the table look-up techniques described above, resulting in $C < N = 256$ (e.g., less than 256 code vectors), about half the amount of memory is used with more consistent relative redundancy.

[0137] Table 1: Experimental results obtained with the implementation of the techniques described herein, applied to data used in the compression of a Kodak dataset with 24 images.

Quant. PDF param.	N	C	Memory (Kbytes)	Relative redundancy (%) @ bit rate				
				0.25 bpp	0.5 bpp	1.0 bpp	1.5 bpp	2.5 bpp
σ	256	256	498.6	754.7	858.7	923.7	951.6	899.6
v	256	256	108.7	0.03	0.02	0.02	0.01	0.01
v	43	43	18.3	0.51	0.37	0.32	0.23	0.23
v	256	43	7.7	0.31	0.30	0.30	0.29	0.29
v	23	23	9.8	1.45	1.15	0.99	0.87	0.76
v	256	23	4.4	0.84	0.89	0.90	0.90	0.93

[0138] As shown above, for image and video coding using neural networks, selecting parameters for entropy coding that are most intuitive and convenient for the designers can work when using high-precision floating point implementation, but can cause serious degradation when the weights and activations of the neural networks are quantized. In fact, analysis of the commonly used Gaussian distributions shows that the worst degradation of compression efficiency (highest relative redundancy) occurs exactly for the most common cases obtained during compression.

[0139] The proposed techniques of this disclosure address the problem by re-defining the parameter used for entropy coding, preserving the training process with a single change of variables, but modifying the quantization to equalize the redundancy increase resulting from quantization. In addition, it is shown that, using the new parameterization, it is possible to reduce the amount of memory used for entropy coding using simple table-look-up for code vector selection.

[0140] FIG. 19 is a flowchart illustrating an example method of the disclosure. The techniques of FIG. 19 may be performed by one or more components of media encoder 200 and/or media decoder 300, including the components shown in the example of FIG. 14. In the example of FIG. 19, media encoder 200 and/or media decoder 300 may be configured to determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream (1900), determine a code vector based on the probability distribution function parameter (1902), and arithmetically code the data element using the code vector (1904).

[0141] Other Aspects of the disclosure are described below.

[0142] Aspect 1 - A method of coding media data, the method comprising: determining a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream; determining a code vector based on the probability distribution function parameter; and entropy coding the data element using the code vector.

[0143] Aspect 2 - The method of Aspect 1, further comprising: quantizing the probability distribution function parameter prior to determining the code vector.

[0144] Aspect 3 - The method of any of Aspects 1-2, wherein determining the code vector comprising: determining a code vector index based on the probability distribution function parameter; and determining the code vector from a table of code vectors based on the code vector index.

[0145] Aspect 4 - The method of any of Aspects 1-3, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

[0146] Aspect 5 - The method of any of Aspects 1-4, further comprising: generating the data element using the neural-based compression technique; and quantizing the data element to create a quantized data element, wherein entropy coding the data element using the code vector comprises entropy encoding the quantized data element using the code vector.

[0147] Aspect 6 - The method of Aspect 5, wherein generating the data element using the neural-based compression technique comprises: processing an image or video picture using an image analysis neural network to generate the data element.

[0148] Aspect 7 - The method of any of Aspects 1-4, wherein entropy coding the data element using the code vector comprises entropy decoding an encoded data element using the code vector to create a quantized data element, the method further comprising: dequantizing the quantized data element to create a reconstructed data element.

[0149] Aspect 8 - The method of Aspect 7, further comprising: processing the reconstructed data element using an image synthesis neural network to reconstruct an image or video picture.

[0150] Aspect 9 - An apparatus configured to code media data, the apparatus comprising: a memory; and one or more processors in communication with the memory, the one or more processors configured to: determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream; determine a code vector based on the probability distribution function parameter; and entropy code the data element using the code vector.

[0151] Aspect 10 - The apparatus of Aspect 9, wherein the one or more processors are further configured to: quantize the probability distribution function parameter prior to determining the code vector.

[0152] Aspect 11 - The apparatus of any of Aspects 9-10, wherein to determine the code vector, the one or more processors are further configured to: determine a code vector index based on the probability distribution function parameter; and determine the code vector from a table of code vectors based on the code vector index.

[0153] Aspect 12 - The apparatus of any of Aspects 9-11, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

[0154] Aspect 13 - The apparatus of any of Aspects 9-11, wherein the one or more processors are further configured to: generate the data element using the neural-based compression technique; and quantize the data element to create a quantized data element, wherein entropy coding the data element using the code vector comprises entropy encoding the quantized data element using the code vector.

[0155] Aspect 14 - The apparatus of Aspect 13, wherein to generate the data element using the neural-based compression technique, the one or more processors are further configured to: process an image or video picture using an image analysis neural network to generate the data element.

[0156] Aspect 15 - The apparatus of Aspect 14, further comprising: a camera configured to capture the image or video picture.

[0157] Aspect 16 - The apparatus of any of Aspects 9-12, wherein entropy coding the data element using the code vector comprises entropy decoding an encoded data element using the code vector to create a quantized data element, and wherein the one or more processors are further configured to: dequantize the quantized data element to create a reconstructed data element.

[0158] Aspect 17 - The apparatus of Aspect 16, wherein the one or more processors are further configured to: process the reconstructed data element using an image synthesis neural network to reconstruct an image or video picture.

[0159] Aspect 18 - The apparatus of Aspect 17, further comprising: a display configured to display the image or video picture.

[0160] Aspect 19 - A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors to: determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream; determine a code vector based on the probability distribution function parameter; and entropy code the data element using the code vector.

[0161] Aspect 20 - The non-transitory computer-readable storage medium of claim 19, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

[0162] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially.

[0163] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0164] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc,

as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0165] Instructions may be executed by one or more processors, such as one or more DSPs, general purpose microprocessors, ASICs, FPGAs, or other equivalent integrated or discrete logic circuitry. Accordingly, the terms “processor” and “processing circuitry,” as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0166] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0167] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of coding media data, the method comprising:
 - determining a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream;
 - determining a code vector based on the probability distribution function parameter; and
 - entropy coding the data element using the code vector.
2. The method of claim 1, further comprising:
 - quantizing the probability distribution function parameter prior to determining the code vector.
3. The method of claim 1, wherein determining the code vector comprises:
 - determining a code vector index based on the probability distribution function parameter; and
 - determining the code vector from a table of code vectors based on the code vector index.
4. The method of claim 1, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

5. The method of claim 1, further comprising:
generating the data element using the neural-based compression technique;
and
quantizing the data element to create a quantized data element,
wherein entropy coding the data element using the code vector comprises
entropy encoding the quantized data element using the code vector.
6. The method of claim 5, wherein generating the data element using the
neural-based compression technique comprises:
processing an image or video picture using an image analysis neural network
to generate the data element.
7. The method of claim 1, wherein entropy coding the data element using the
code vector comprises entropy decoding an encoded data element using the code
vector to create a quantized data element, the method further comprising:
dequantizing the quantized data element to create a reconstructed data
element.
8. The method of claim 7, further comprising:
processing the reconstructed data element using an image synthesis neural
network to reconstruct an image or video picture.

9. An apparatus configured to code media data, the apparatus comprising:
a memory; and
one or more processors in communication with the memory, the one or more processors configured to:
- determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream;
 - determine a code vector based on the probability distribution function parameter; and
 - entropy code the data element using the code vector.
10. The apparatus of claim 9, wherein the one or more processors are further configured to:
- quantize the probability distribution function parameter prior to determining the code vector.
11. The apparatus of claim 9, wherein to determine the code vector, the one or more processors are further configured to:
- determine a code vector index based on the probability distribution function parameter; and
 - determine the code vector from a table of code vectors based on the code vector index.
12. The apparatus of claim 9, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

13. The apparatus of claim 9, wherein the one or more processors are further configured to:
- generate the data element using the neural-based compression technique; and
 - quantize the data element to create a quantized data element,
 - wherein entropy coding the data element using the code vector comprises entropy encoding the quantized data element using the code vector.
14. The apparatus of claim 13, wherein to generate the data element using the neural-based compression technique, the one or more processors are further configured to:
- process an image or video picture using an image analysis neural network to generate the data element.
15. The apparatus of claim 14, further comprising:
- a camera configured to capture the image or video picture.
16. The apparatus of claim 9, wherein entropy coding the data element using the code vector comprises entropy decoding an encoded data element using the code vector to create a quantized data element, and wherein the one or more processors are further configured to:
- dequantize the quantized data element to create a reconstructed data element.
17. The apparatus of claim 16, wherein the one or more processors are further configured to:
- process the reconstructed data element using an image synthesis neural network to reconstruct an image or video picture.
18. The apparatus of claim 17, further comprising:
- a display configured to display the image or video picture.

19. A non-transitory computer-readable storage medium storing instructions that, when executed, cause one or more processors to:

determine a probability distribution function parameter for a data element of a data stream coded by a neural-based media compression technique, wherein the probability distribution function parameter is a logarithmic function of a standard deviation of a probability distribution function of the data stream;

determine a code vector based on the probability distribution function parameter; and

entropy code the data element using the code vector.

20. The non-transitory computer-readable storage medium of claim 19, wherein the probability distribution function parameter is v , the standard deviation of the probability distribution function is σ , a minimum standard deviation is $\sigma_{\min}$, a maximum standard deviation is $\sigma_{\max}$, and wherein v is defined as:

$$v = \frac{\ln(\sigma) - \ln(\sigma_{\min})}{\ln(\sigma_{\max}) - \ln(\sigma_{\min})}.$$

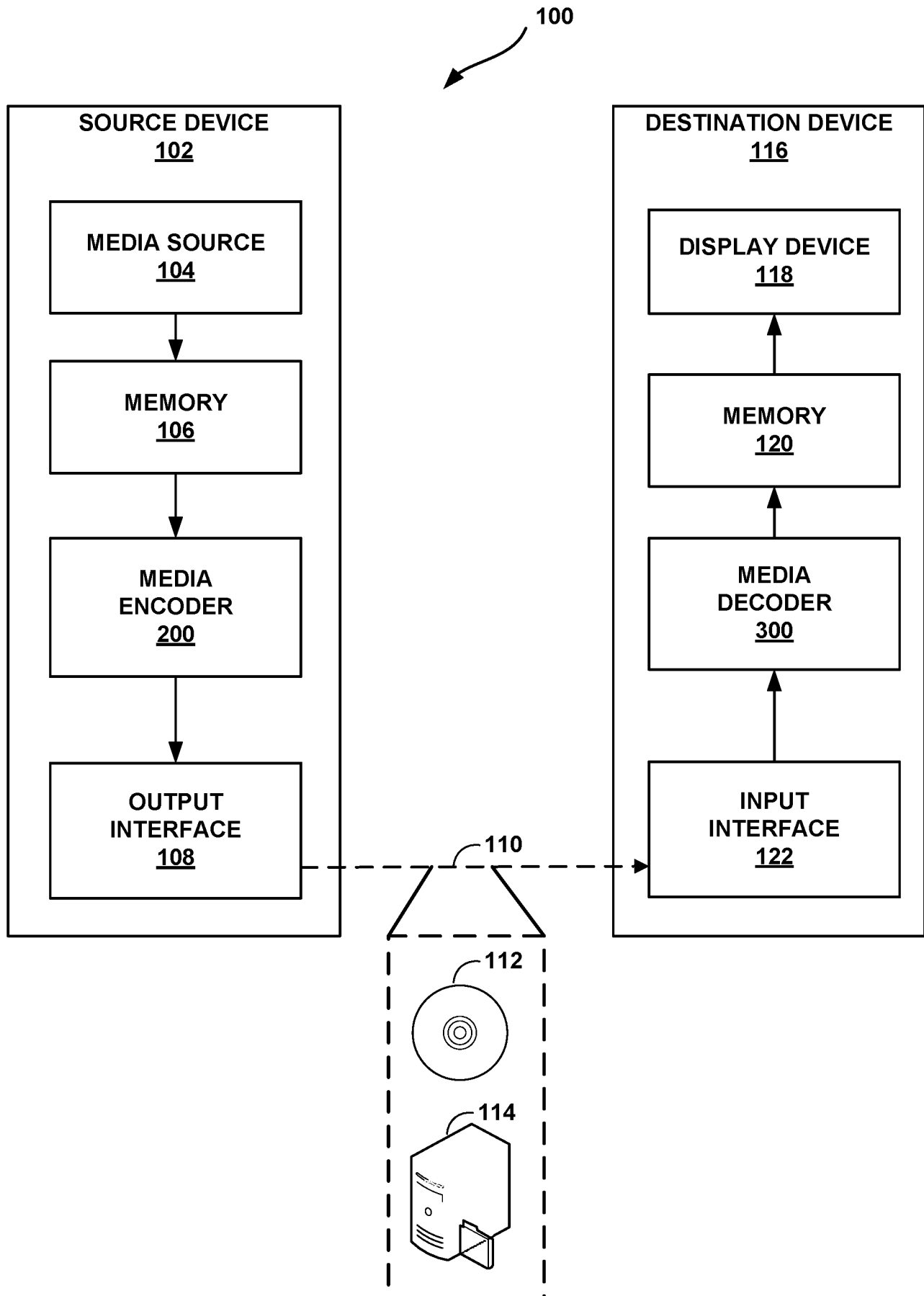


FIG. 1

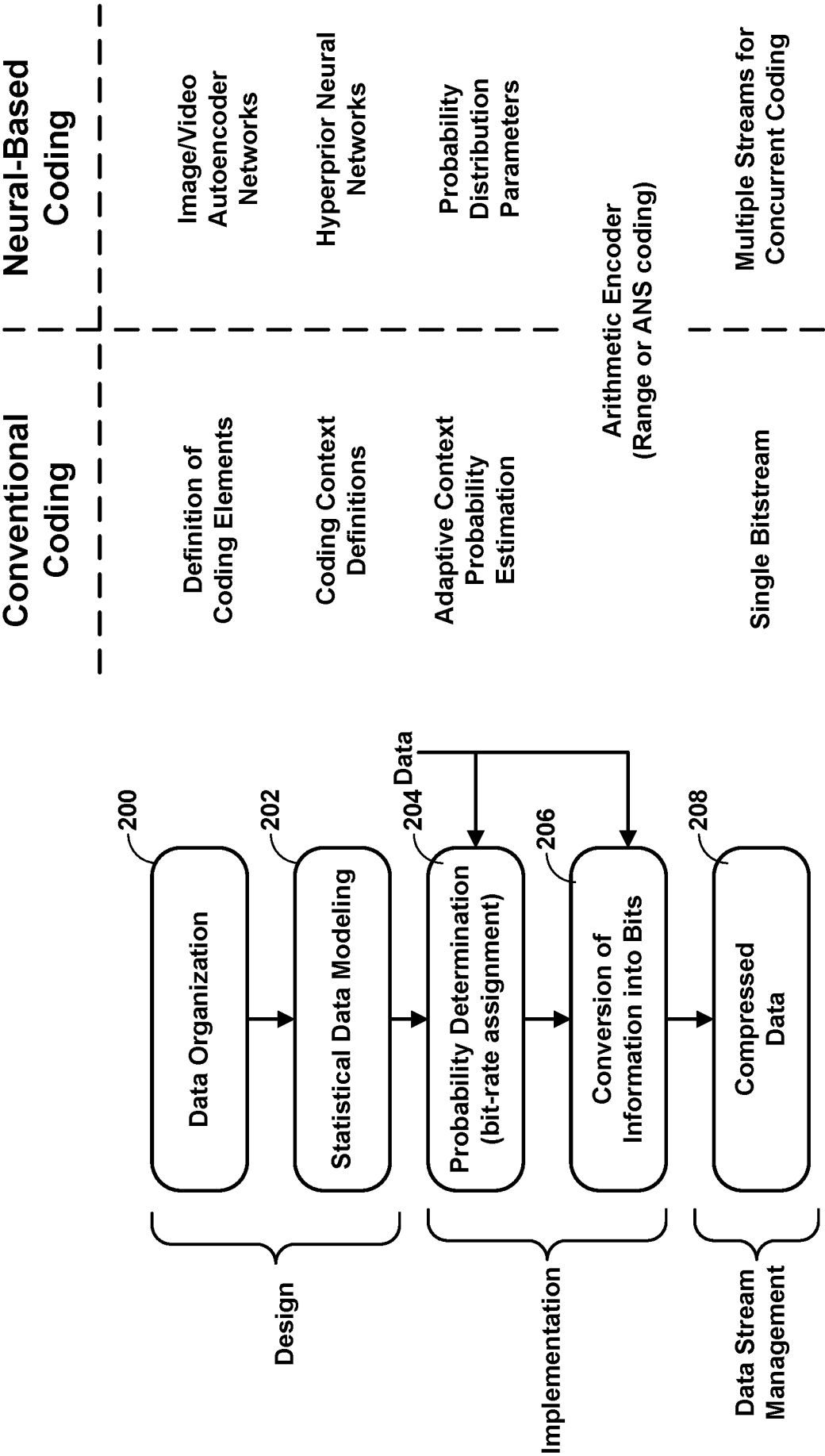


FIG. 2

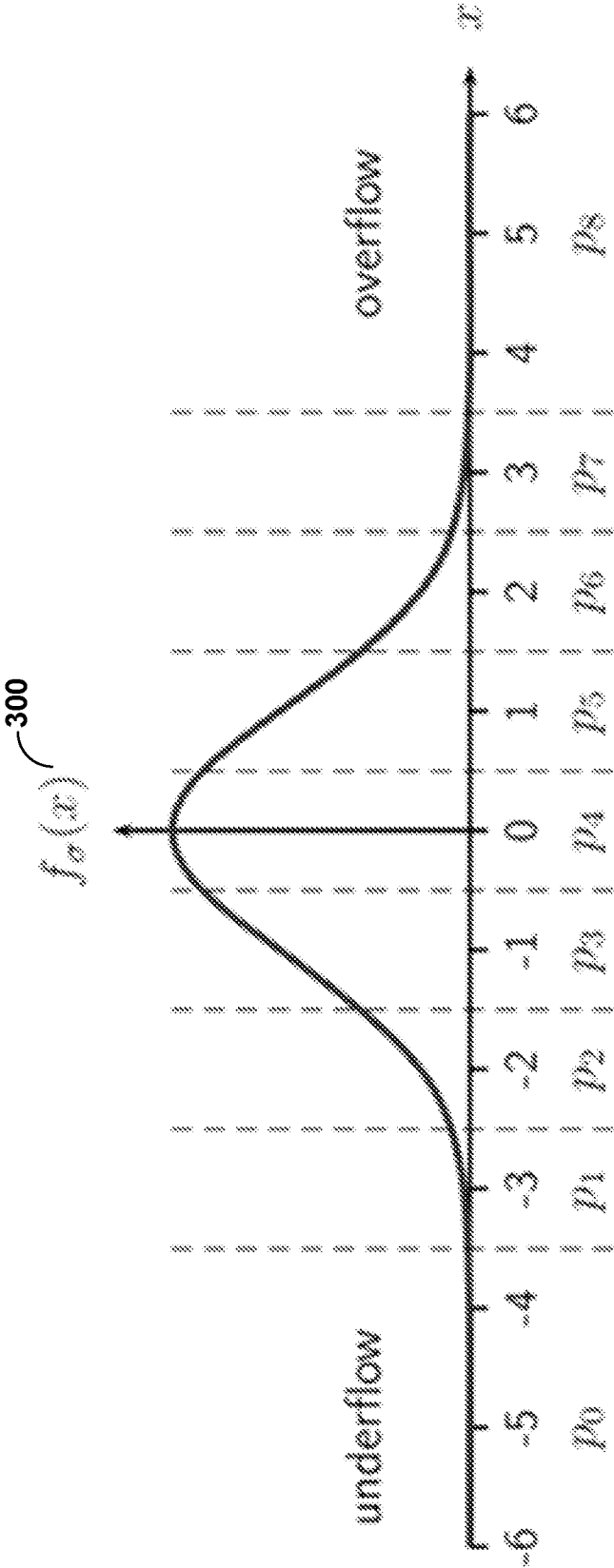


FIG. 3

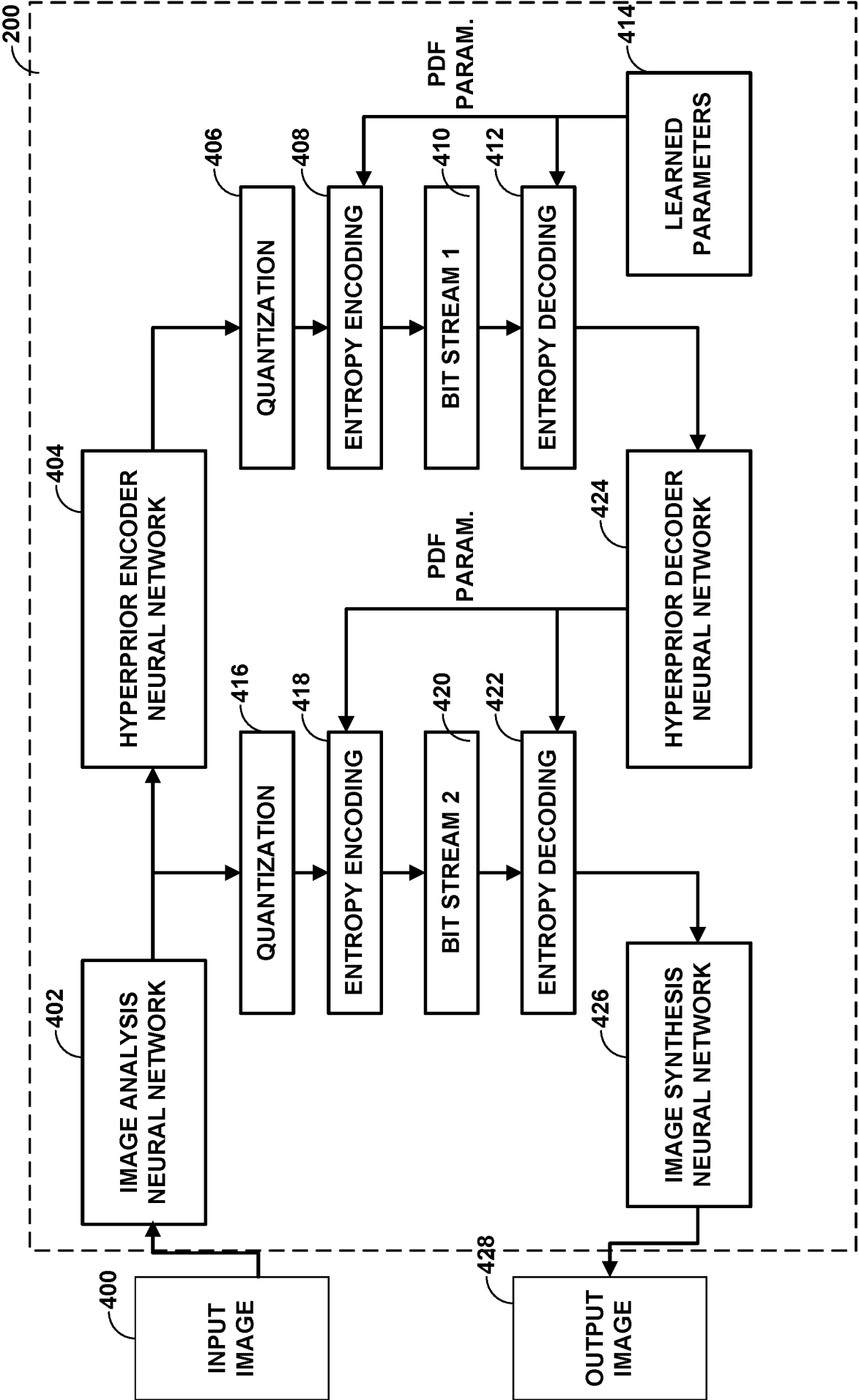


FIG. 4

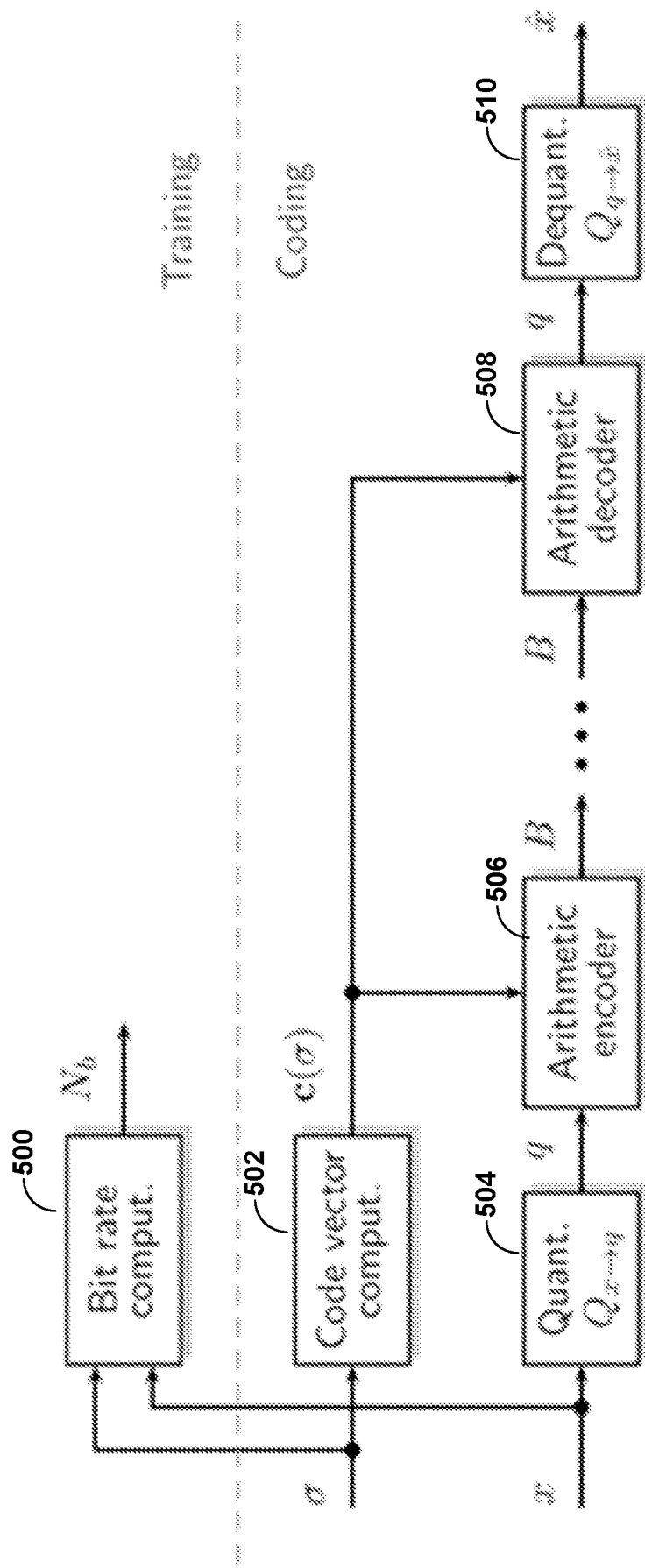


FIG. 5

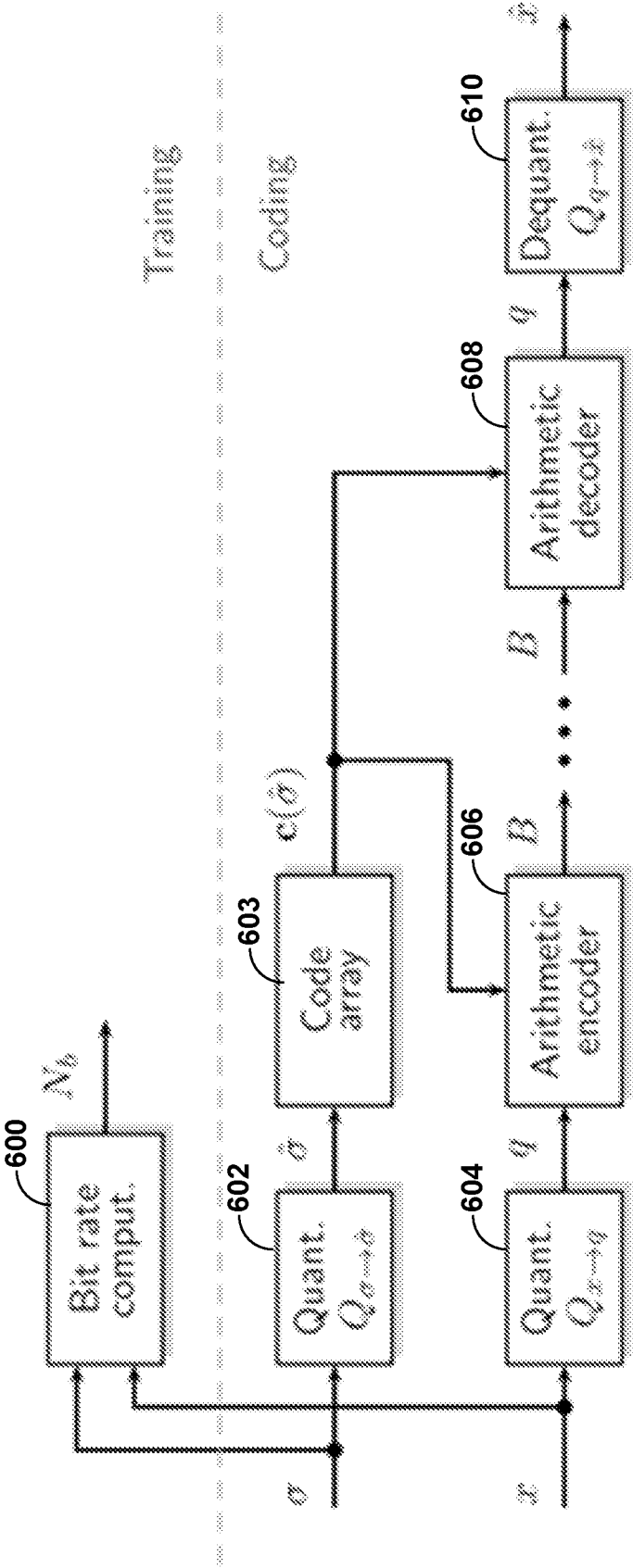


FIG. 6

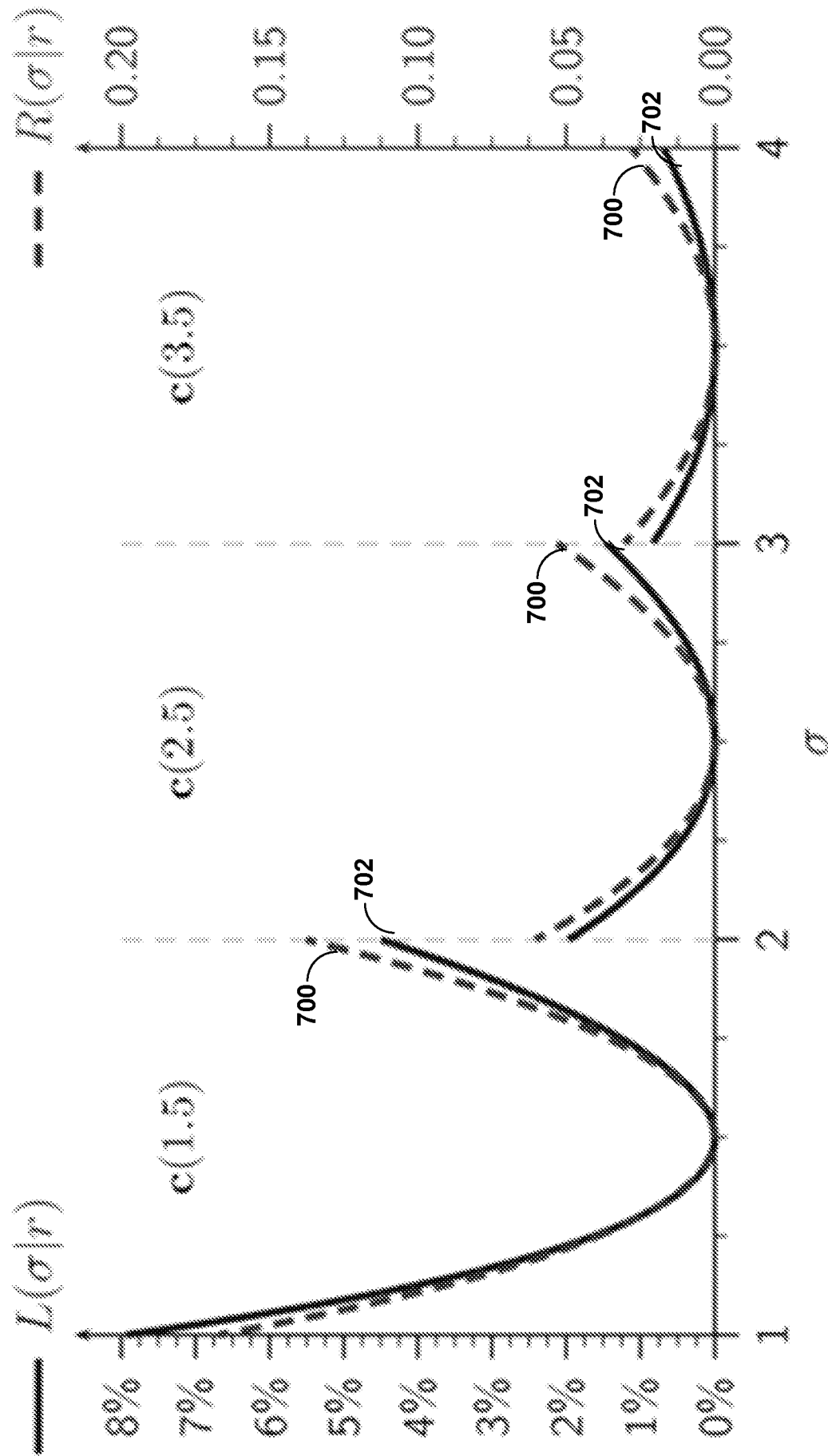


FIG. 7

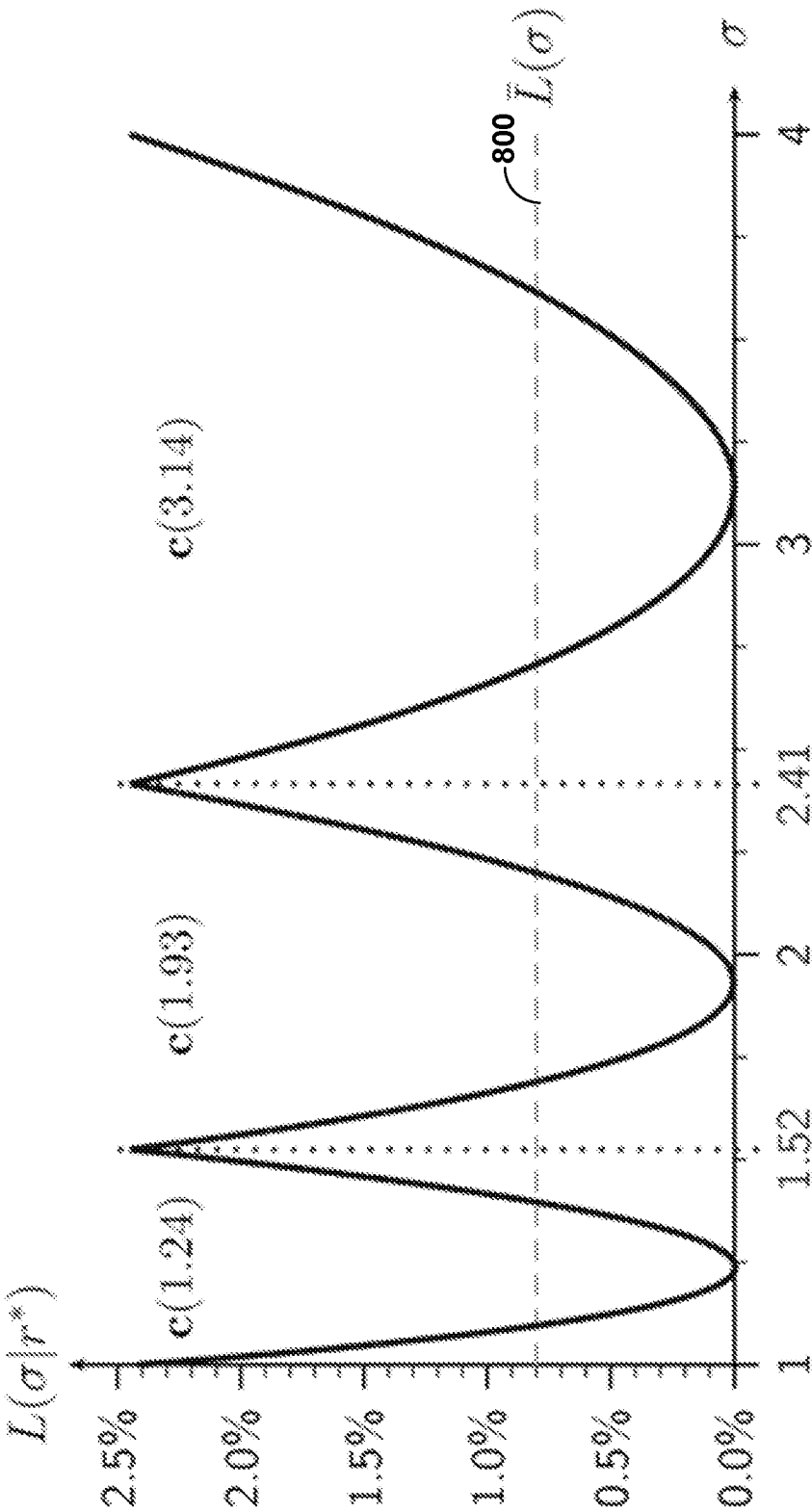


FIG. 8

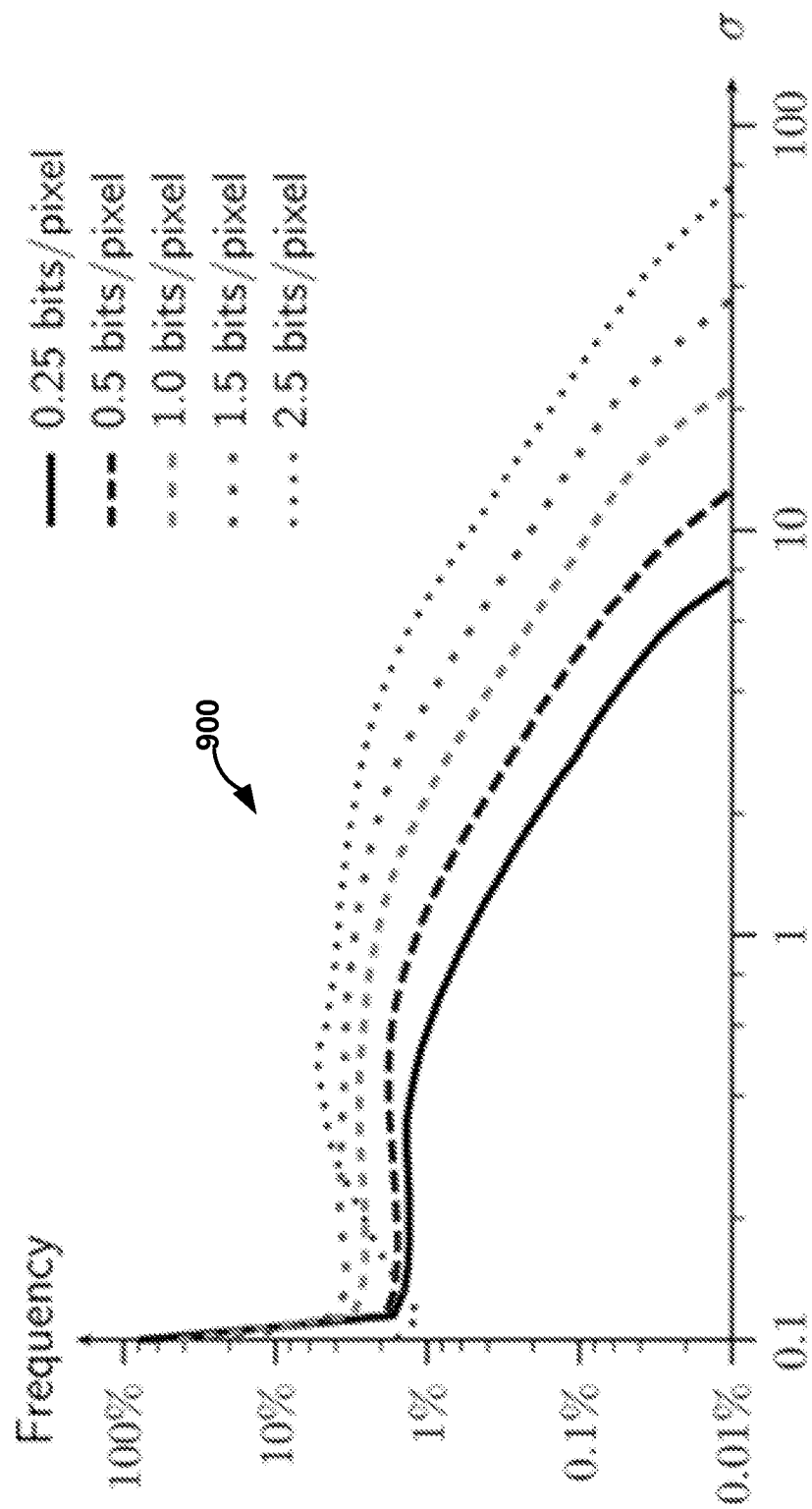


FIG. 9

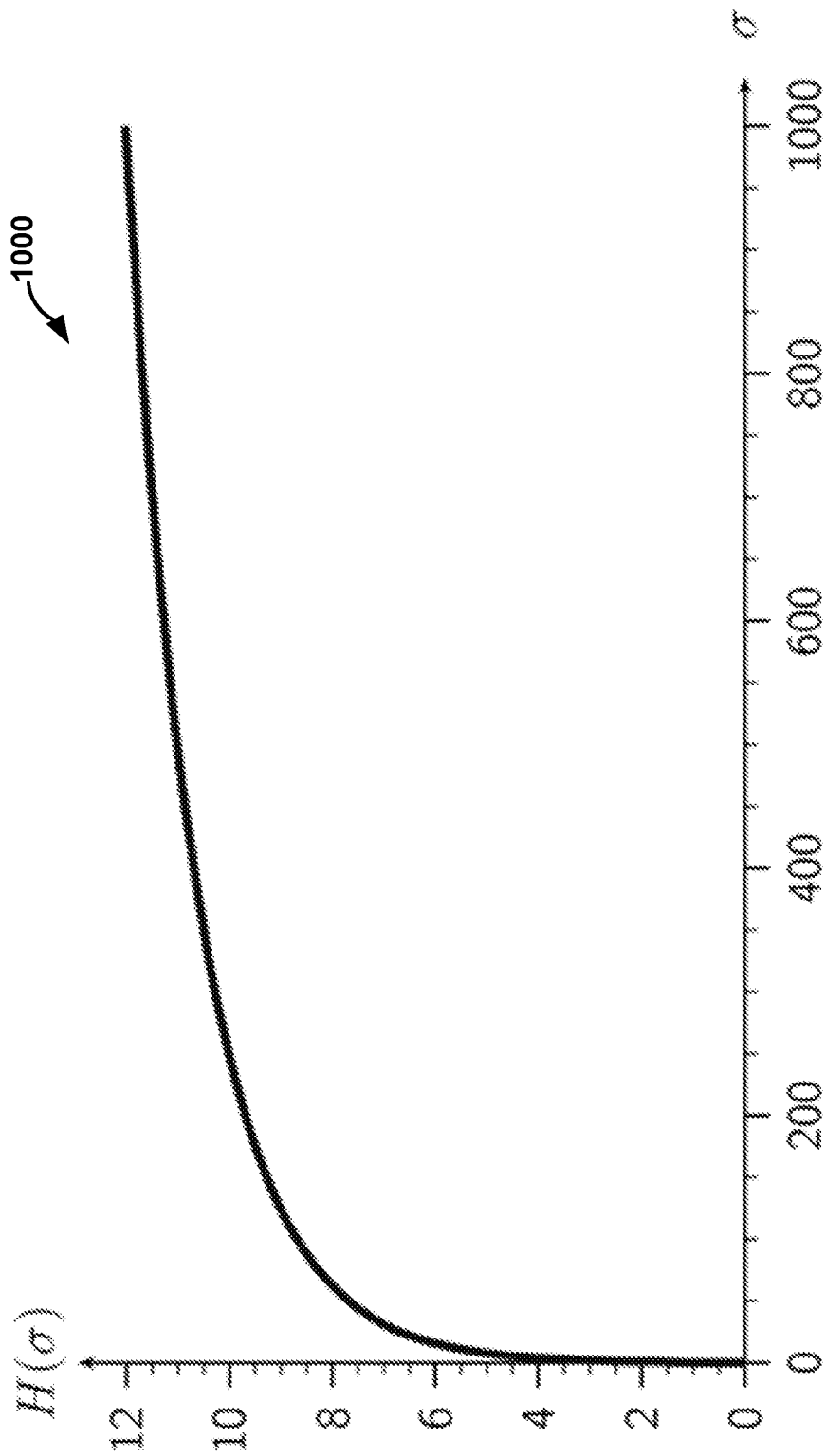


FIG. 10

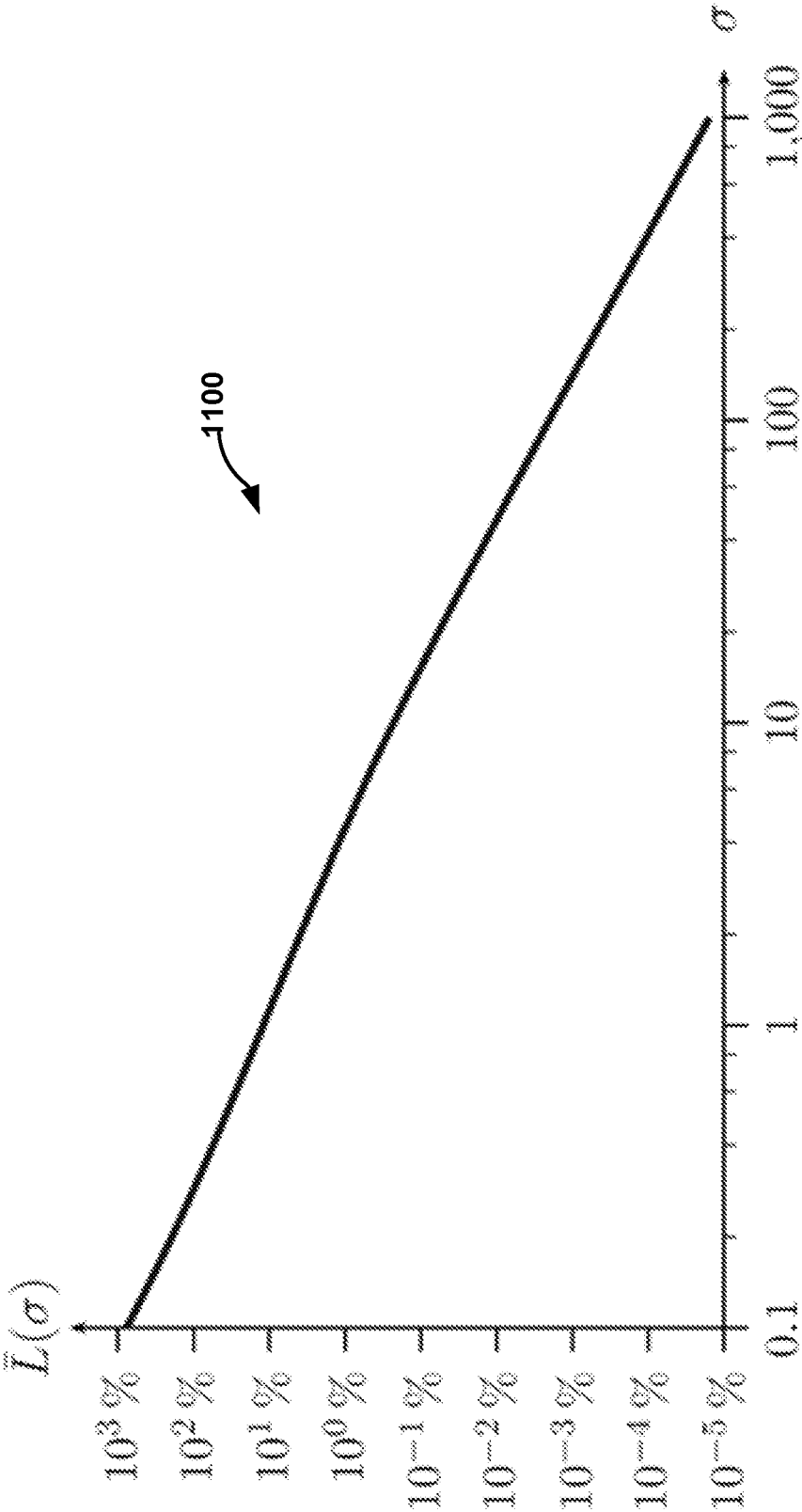


FIG. 11

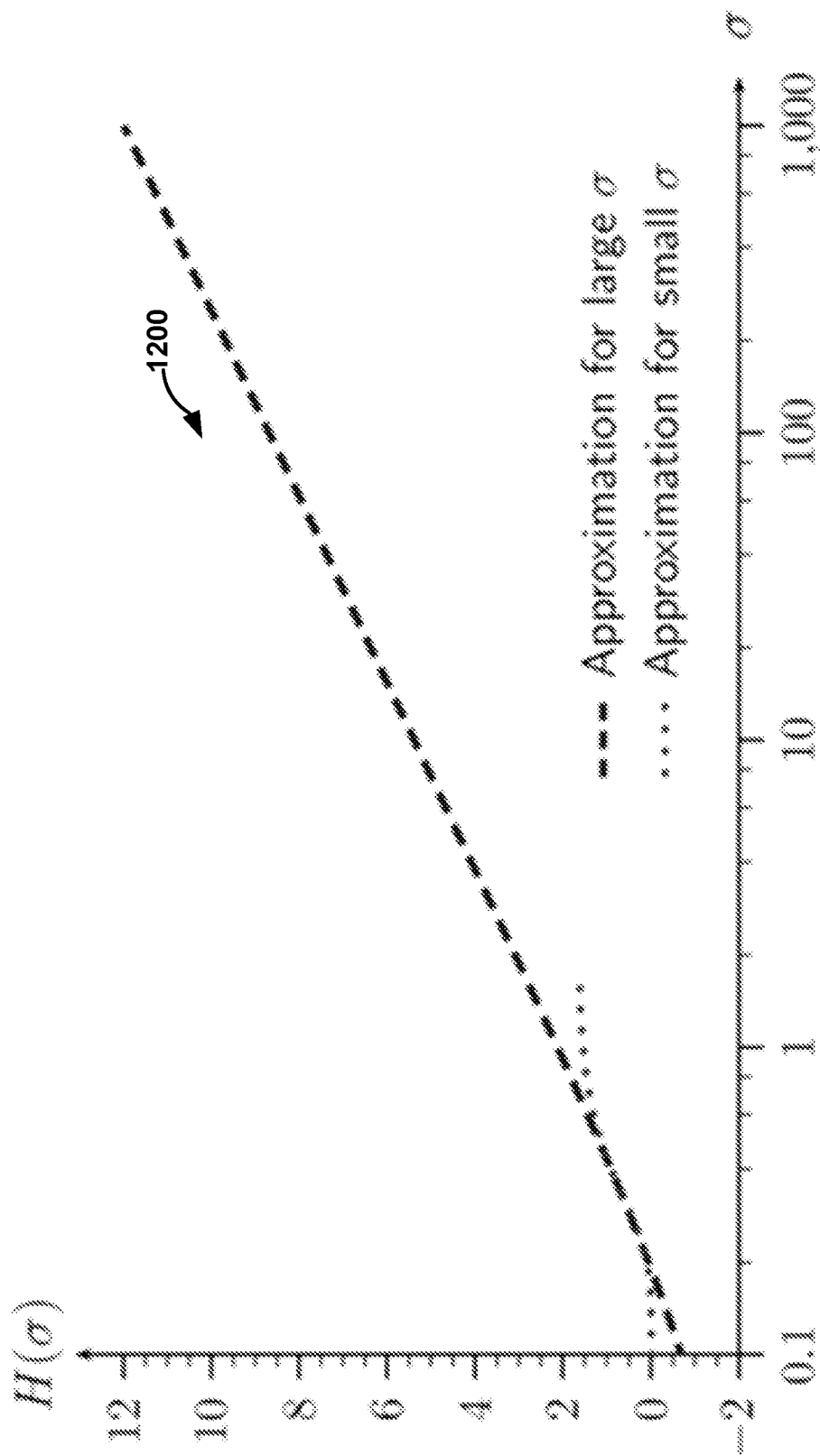


FIG. 12

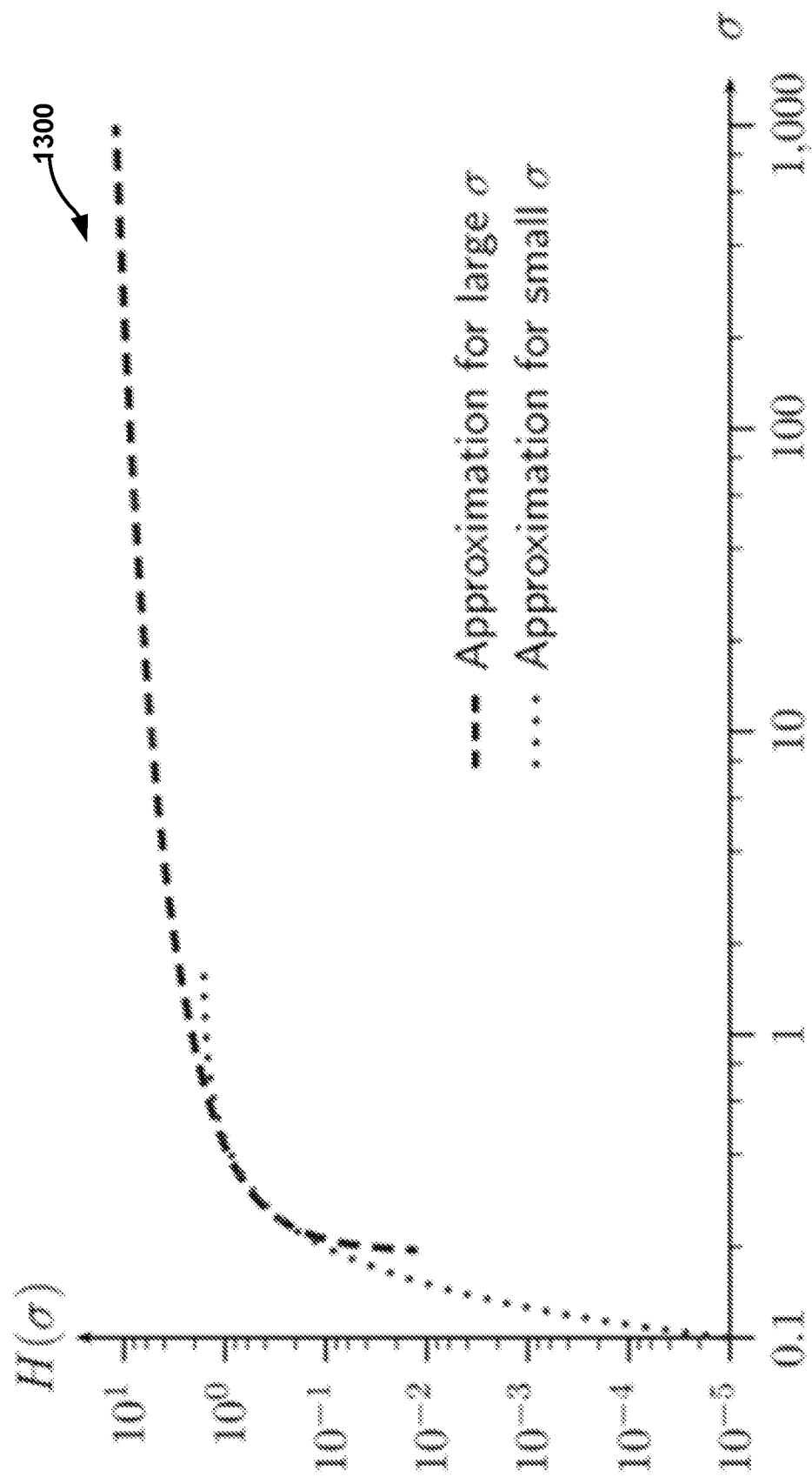


FIG. 13

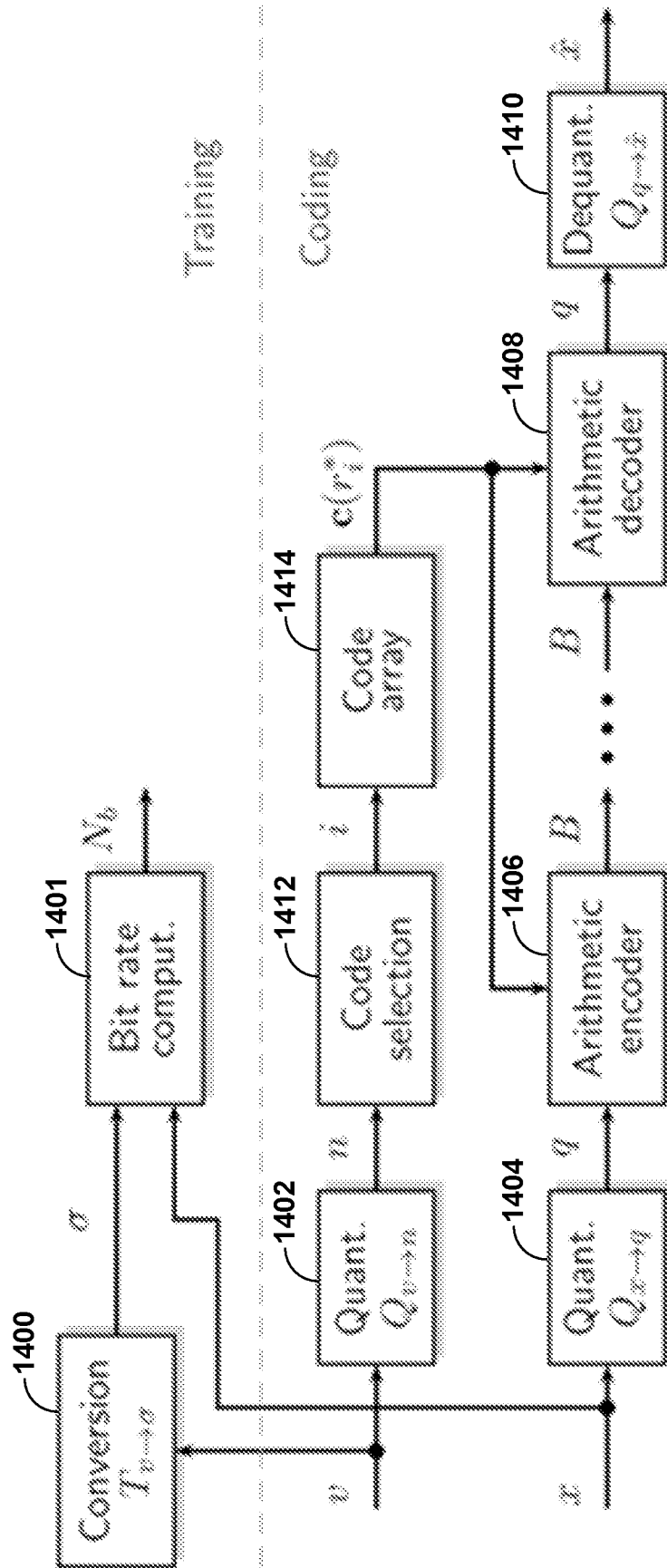


FIG. 14

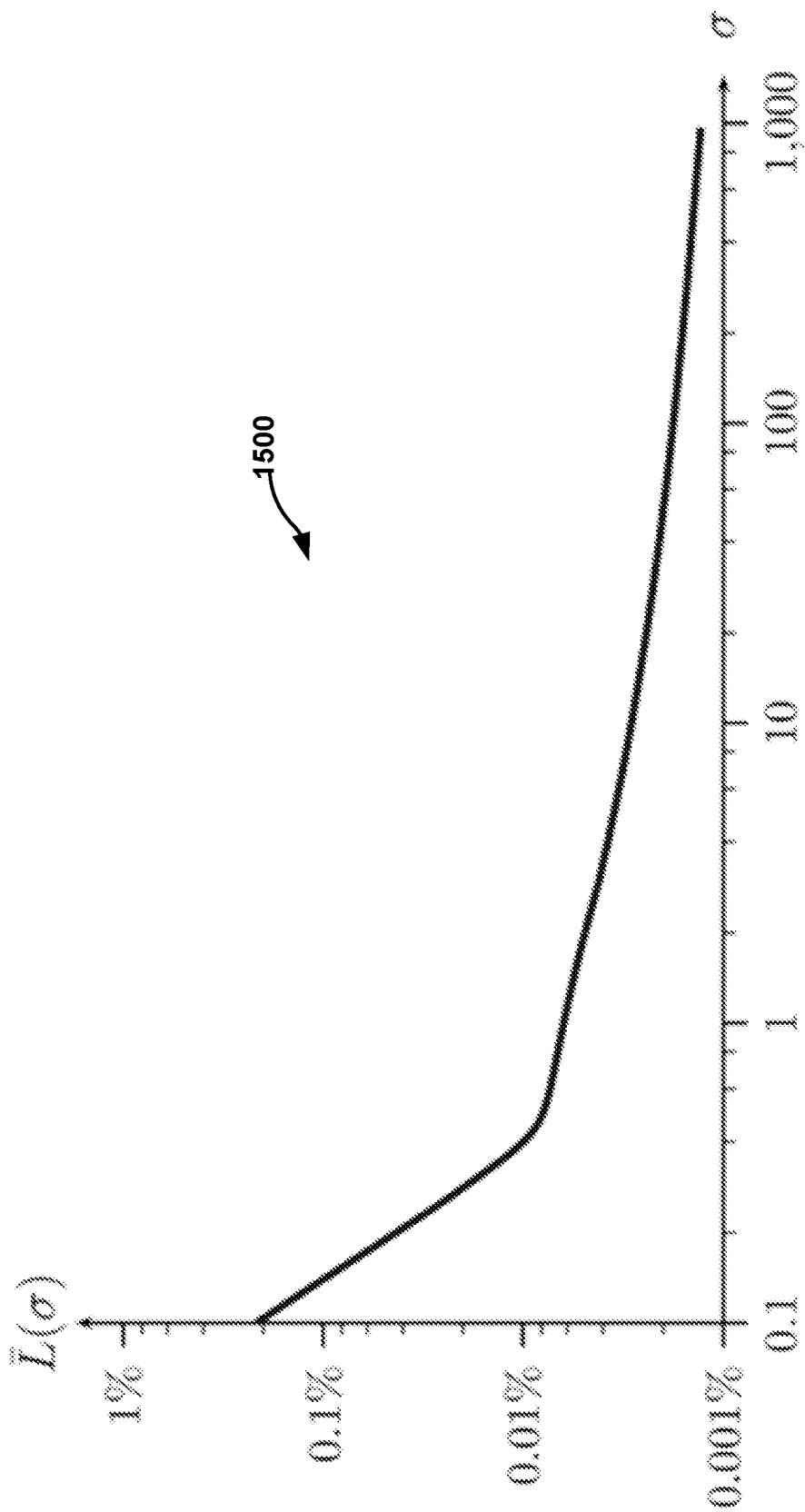


FIG. 15

1600
↙

```

const uint8_t code_index[256] = {
    0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13,
    14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
    28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
    42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
    56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
    70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
    84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
    98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111,
    112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
    126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139,
    140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,
    154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
    168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
    182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
    196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209,
    210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223,
    224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
    238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251,
    252, 253, 254, 255 };

```

FIG. 16

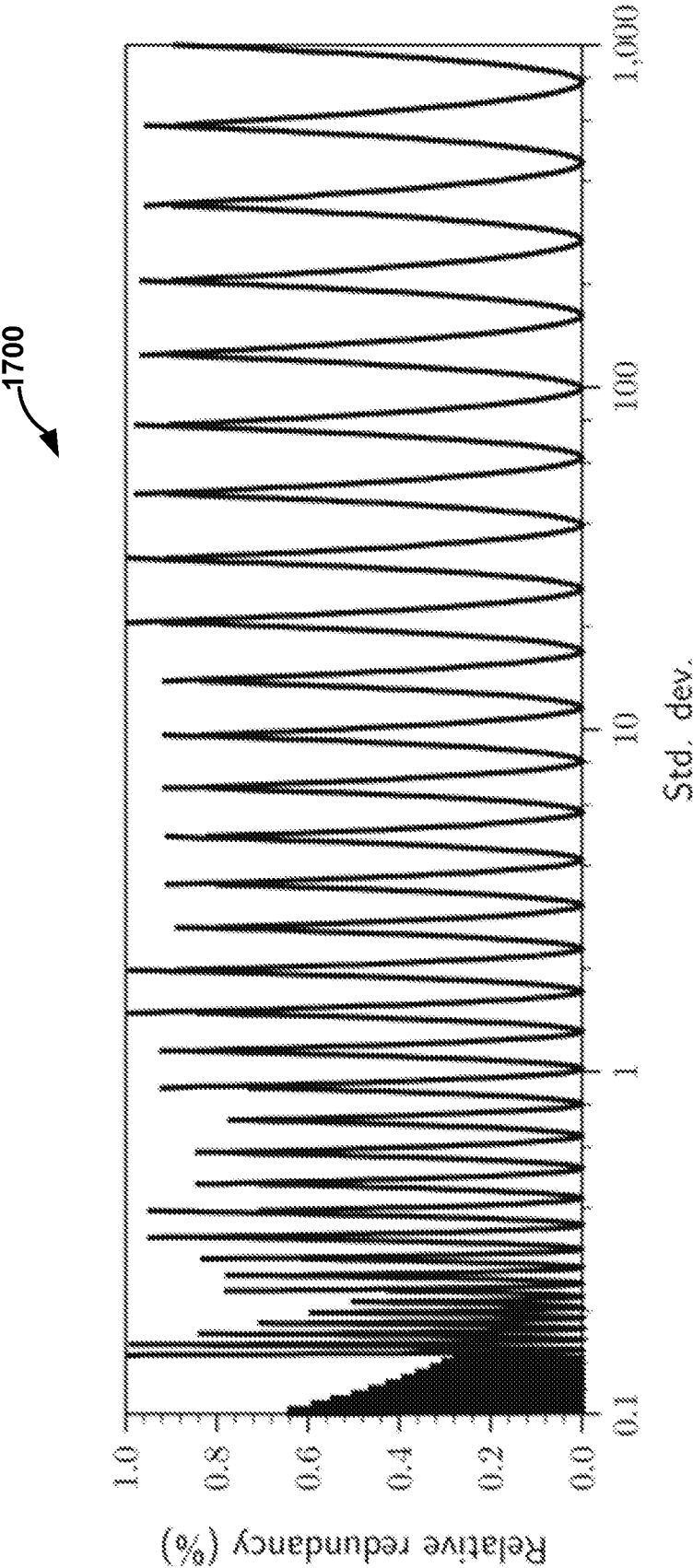


FIG. 17

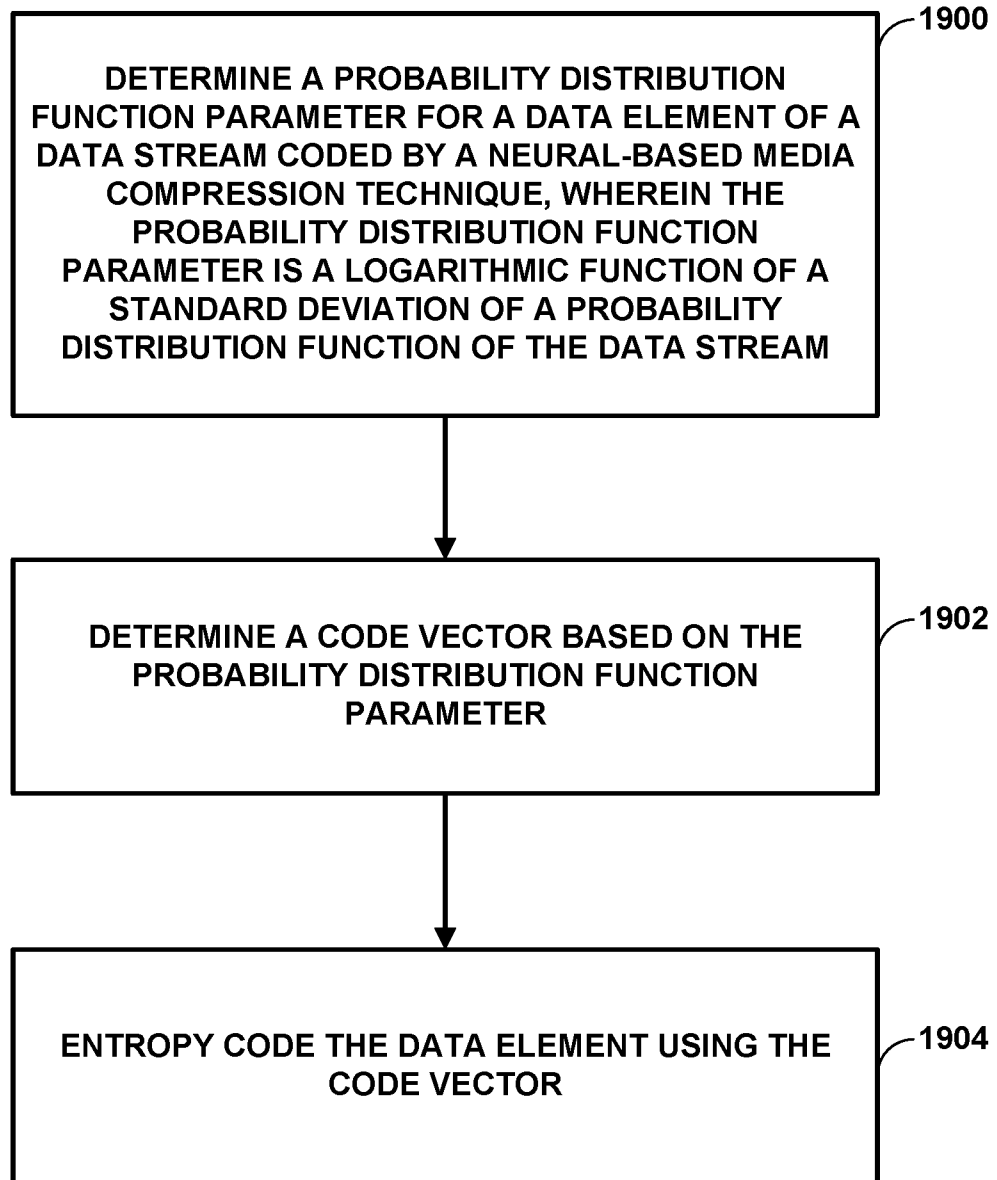


FIG. 19

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2023/060460

A. CLASSIFICATION OF SUBJECT MATTER INV. H04N19/91 H04N19/94 H04N19/463 ADD. 		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) H04N		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EPO-Internal, INSPEC, WPI Data		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Johannes Ballé ET AL: "Variational image compression with a scale hyperprior", / 1 May 2018 (2018-05-01), XP055632204, Retrieved from the Internet: URL:https://arxiv.org/pdf/1802.01436.pdf [retrieved on 2019-10-15]	1,2, 4-10, 12-20
A	the whole document ----- -/--	3,11
☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents : "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family		
Date of the actual completion of the international search	Date of mailing of the international search report	
8 March 2023	28/03/2023	
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Cyranka, Oliver	

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2023/060460

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BRUMMER BENOIT ET AL: "End-to-end optimized image compression with competition of prior distributions", 2021 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS (CVPRW), IEEE, 19 June 2021 (2021-06-19), pages 1890-1894, XP033967894, DOI: 10.1109/CVPRW53098.2021.00212 [retrieved on 2021-08-25]	1-3, 5, 7, 9-11, 13, 16, 19
A	the whole document	4, 6, 8, 12, 14, 15, 17, 18, 20
A	----- ZHAO JING ET AL: "A Universal Encoder Rate Distortion Optimization Framework for Learned Compression", 2021 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS (CVPRW), IEEE, 19 June 2021 (2021-06-19), pages 1880-1884, XP033967384, DOI: 10.1109/CVPRW53098.2021.00210 [retrieved on 2021-08-25] the whole document	1-20
T	----- AMIR SAID ET AL: "Optimized learned entropy coding parameters for practical neural-based image and video compression", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 20 January 2023 (2023-01-20), XP091419131, DOI: 10.1109/ICIP46576.2022.9897505 the whole document -----	1-20