CONTENT-ADAPTIVE ONLINE TRAINING METHOD AND APPARATUS FOR POST-FILTERING

INCORPORATION BY REFERENCE

[0001] This present application claims the benefit of priority to U.S. Patent Application No. 17/749,641, “CONTENT-ADAPTIVE ONLINE TRAINING METHOD AND APPARATUS FOR POST-FILTERING” filed on May 20, 2022, which claims the benefit of priority to U.S. Provisional Application No. 63/194,057, "Content-Adaptive Online Training Method for Post-Filtering" filed on May 27, 2021. The disclosures of the prior applications are hereby incorporated by reference in their entirety.

TECHNICAL FIELD

[0002] The present disclosure describes embodiments generally related to video coding.

BACKGROUND

[0003] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0004] Video coding and decoding can be performed using inter-picture prediction with motion compensation. Uncompressed digital image and/or video can include a series of pictures, each picture having a spatial dimension of, for example, 1920 x 1080 luminance samples and associated chrominance samples. The series of pictures can have a fixed or variable picture rate (informally also known as frame rate), of, for example 60 pictures per second or 60 Hz. Uncompressed image and/or video has specific bitrate requirements. For example, 1080p60 4:2:0 video at 8 bit per sample (1920x1080 luminance sample resolution at 60 Hz frame rate) requires close to 1.5 Gbit/s bandwidth. An hour of such video requires more than 600 GBytes of storage space.

[0005] One purpose of video coding and decoding can be the reduction of redundancy in the input image and/or video signal, through compression. Compression can help reduce the aforementioned bandwidth and/or storage space requirements, in some cases by two orders of magnitude or more. Although the descriptions herein uses video encoding/decoding as illustrative examples, the same techniques can be applied to image encoding/decoding in similar fashion without departing from the spirits of the present disclosure. Both lossless

compression and lossy compression, as well as a combination thereof can be employed. Lossless compression refers to techniques where an exact copy of the original signal can be reconstructed from the compressed original signal. When using lossy compression, the reconstructed signal may not be identical to the original signal, but the distortion between original and reconstructed signals is small enough to make the reconstructed signal useful for the intended application. In the case of video, lossy compression is widely employed. The amount of distortion tolerated depends on the application; for example, users of certain consumer streaming applications may tolerate higher distortion than users of television distribution applications. The compression ratio achievable can reflect that: higher allowable/tolerable distortion can yield higher compression ratios.

[0006] A video encoder and decoder can utilize techniques from several broad categories, including, for example, motion compensation, transform, quantization, and entropy coding.

[0007] Video codec technologies can include techniques known as intra coding. In intra coding, sample values are represented without reference to samples or other data from previously reconstructed reference pictures. In some video codecs, the picture is spatially subdivided into blocks of samples. When all blocks of samples are coded in intra mode, that picture can be an intra picture. Intra pictures and their derivations such as independent decoder refresh pictures, can be used to reset the decoder state and can, therefore, be used as the first picture in a coded video bitstream and a video session, or as a still image. The samples of an intra block can be exposed to a transform, and the transform coefficients can be quantized before entropy coding. Intra prediction can be a technique that minimizes sample values in the pre-transform domain. In some cases, the smaller the DC value after a transform is, and the smaller the AC coefficients are, the fewer the bits that are required at a given quantization step size to represent the block after entropy coding.

[0008] Traditional intra coding such as known from, for example MPEG-2 generation coding technologies, does not use intra prediction. However, some newer video compression technologies include techniques that attempt, from, for example, surrounding sample data and/or metadata obtained during the encoding and/or decoding of spatially neighboring, and preceding in decoding order, blocks of data. Such techniques are henceforth called “intra prediction” techniques. Note that in at least some cases, intra prediction is using reference data only from the current picture under reconstruction and not from reference pictures.

[0009] There can be many different forms of intra prediction. When more than one of such techniques can be used in a given video coding technology, the technique in use can be coded in an intra prediction mode. In certain cases, modes can have submodes and/or parameters, and those can be coded individually or included in the mode codeword. Which codeword to use for a given mode, submode, and/or parameter combination can have an impact in the coding efficiency gain through intra prediction, and so can the entropy coding technology used to translate the codewords into a bitstream.

[0010] A certain mode of intra prediction was introduced with H.264, refined in H.265, and further refined in newer coding technologies such as joint exploration model (JEM), versatile video coding (VVC), and benchmark set (BMS). A predictor block can be formed using neighboring sample values belonging to already available samples. Sample values of neighboring samples are copied into the predictor block according to a direction. A reference to the direction in use can be coded in the bitstream or may itself be predicted.

[0011] Referring to FIG. 1A, depicted in the lower right is a subset of nine predictor directions known from H.265's 33 possible predictor directions (corresponding to the 33 angular modes of the 35 intra modes). The point where the arrows converge (101) represents the sample being predicted. The arrows represent the direction from which the sample is being predicted. For example, arrow (102) indicates that sample (101) is predicted from a sample or samples to the upper right, at a 45 degree angle from the horizontal. Similarly, arrow (103) indicates that sample (101) is predicted from a sample or samples to the lower left of sample (101), in a 22.5 degree angle from the horizontal.

[0012] Still referring to FIG. 1A, on the top left there is depicted a square block (104) of 4 x 4 samples (indicated by a dashed, boldface line). The square block (104) includes 16 samples, each labelled with an "S", its position in the Y dimension (e.g., row index) and its position in the X dimension (e.g., column index). For example, sample S21 is the second sample in the Y dimension (from the top) and the first (from the left) sample in the X dimension. Similarly, sample S44 is the fourth sample in block (104) in both the Y and X dimensions. As the block is 4 x 4 samples in size, S44 is at the bottom right. Further shown are reference samples that follow a similar numbering scheme. A reference sample is labelled with an R, its Y position (e.g., row index) and X position (column index) relative to block (104). In both H.264 and H.265, prediction samples neighbor the block under reconstruction; therefore no negative values need to be used.

[0013] Intra picture prediction can work by copying reference sample values from the neighboring samples as appropriated by the signaled prediction direction. For example, assume the coded video bitstream includes signaling that, for this block, indicates a prediction direction consistent with arrow (102)—that is, samples are predicted from a prediction sample or samples to the upper right, at a 45 degree angle from the horizontal. In that case, samples S41, S32, S23, and S14 are predicted from the same reference sample R05. Sample S44 is then predicted from reference sample R08.

[0014] In certain cases, the values of multiple reference samples may be combined, for example through interpolation, in order to calculate a reference sample; especially when the directions are not evenly divisible by 45 degrees.

[0015] The number of possible directions has increased as video coding technology has developed. In H.264 (year 2003), nine different direction could be represented. That increased to 33 in H.265 (year 2013), and JEM/VVC/BMS, at the time of disclosure, can support up to 65 directions. Experiments have been conducted to identify the most likely directions, and certain techniques in the entropy coding are used to represent those likely directions in a small number of bits, accepting a certain penalty for less likely directions. Further, the directions themselves can sometimes be predicted from neighboring directions used in neighboring, already decoded, blocks.

[0016] FIG. 1B shows a schematic (110) that depicts 65 intra prediction directions according to JEM to illustrate the increasing number of prediction directions over time.

[0017] The mapping of intra prediction directions bits in the coded video bitstream that represent the direction can be different from video coding technology to video coding technology; and can range, for example, from simple direct mappings of prediction direction to intra prediction mode, to codewords, to complex adaptive schemes involving most probable modes, and similar techniques. In all cases, however, there can be certain directions that are statistically less likely to occur in video content than certain other directions. As the goal of video compression is the reduction of redundancy, those less likely directions will, in a well working video coding technology, be represented by a larger number of bits than more likely directions.

[0018] Motion compensation can be a lossy compression technique and can relate to techniques where a block of sample data from a previously reconstructed picture or part thereof (reference picture), after being spatially shifted in a direction indicated by a motion vector (MV henceforth), is used for the prediction of a newly reconstructed picture or picture

part. In some cases, the reference picture can be the same as the picture currently under reconstruction. MVs can have two dimensions X and Y, or three dimensions, the third being an indication of the reference picture in use (the latter, indirectly, can be a time dimension).

[0019] In some video compression techniques, an MV applicable to a certain area of sample data can be predicted from other MVs, for example from those related to another area of sample data spatially adjacent to the area under reconstruction, and preceding that MV in decoding order. Doing so can substantially reduce the amount of data required for coding the MV, thereby removing redundancy and increasing compression. MV prediction can work effectively, for example, because when coding an input video signal derived from a camera (known as natural video) there is a statistical likelihood that areas larger than the area to which a single MV is applicable move in a similar direction and, therefore, can in some cases be predicted using a similar motion vector derived from MVs of neighboring area. That results in the MV found for a given area to be similar or the same as the MV predicted from the surrounding MVs, and that in turn can be represented, after entropy coding, in a smaller number of bits than what would be used if coding the MV directly. In some cases, MV prediction can be an example of lossless compression of a signal (namely: the MVs) derived from the original signal (namely: the sample stream). In other cases, MV prediction itself can be lossy, for example because of rounding errors when calculating a predictor from several surrounding MVs.

[0020] Various MV prediction mechanisms are described in H.265/HEVC (ITU-T Rec. H.265, "High Efficiency Video Coding", December 2016). Out of the many MV prediction mechanisms that H.265 offers, described here is a technique henceforth referred to as "spatial merge".

[0021] Referring to FIG. 2, a current block (201) comprises samples that have been found by the encoder during the motion search process to be predictable from a previous block of the same size that has been spatially shifted. Instead of coding that MV directly, the MV can be derived from metadata associated with one or more reference pictures, for example from the most recent (in decoding order) reference picture, using the MV associated with either one of five surrounding samples, denoted A0, A1, and B0, B1, B2 (202 through 206, respectively). In H.265, the MV prediction can use predictors from the same reference picture that the neighboring block is using.

SUMMARY

[0022] Aspects of the disclosure provide methods and apparatuses for video encoding and decoding. In some examples, an apparatus for video decoding includes processing circuitry. The processing circuitry is configured to receive an image or video comprising one or more blocks. The processing circuitry can decode a first post-filtering parameter in the image or video corresponding to the one or more blocks to be reconstructed. The first post-filtering parameter applies to at least one of the one or more blocks, and the first post-filtering parameter has been updated by a post-filtering module in a post-filtering neural network (NN) that is trained based on a training dataset. The processing circuitry can determine the post-filtering NN in a video decoder corresponding to the one or more blocks based on the first post-filtering parameter. The processing circuitry can decode the one or more blocks based on the determined post-filtering NN corresponding to the one or more blocks.

[0023] In an embodiment, the processing circuitry decodes a second post-filtering parameter in the image or video corresponding to the one or more blocks. The processing circuitry can determine the post-filtering NN further based on the second post-filtering parameter. The second post-filtering parameter applies to a second block in the one or more blocks. The second block is different from the at least one of the one or more blocks. The second post-filtering parameter has been updated by the post-filtering module in the post-filtering NN.

[0024] In an embodiment, the first post-filtering parameter corresponds to a second image to be reconstructed. The processing circuitry can decode the second image based on the determined post-filtering NN.

[0025] In an embodiment, the first post-filtering parameter is different from the second post-filtering parameter. The first post-filtering parameter is adaptive to content of a first block of the one or more blocks, and the second post-filtering parameter is adaptive to content of the second block.

[0026] In an embodiment, the first post-filtering parameter is updated based on a bias term or a weight coefficient in the post-filtering NN.

[0027] In an embodiment, the post-filtering NN is configured with initial parameters. The processing circuitry updates at least one of the initial parameters with the first post-filtering parameter.

[0028] In an embodiment, coding information corresponding to the one or more blocks indicates a difference between the first post-filtering parameter and the one of the

initial parameters. The processing circuitry can determine the first post-filtering parameter according to a sum of the difference and the one of the initial parameters.

[0029] In an embodiment, the first post-filtering parameter is updated in (i) a single layer of the post-filtering NN, (ii) multiple layers of the post-filtering NN, or (iii) all layers of the post-filtering NN.

[0030] In an example, a number of layers in the post-filtering NN is dependent on a step size or a number of steps corresponding to different blocks in the one or more blocks.

[0031] Aspects of the disclosure also provide a non-transitory computer-readable storage medium storing a program executable by at least one processor to perform the methods for video decoding.

BRIEF DESCRIPTION OF THE DRAWINGS

[0032] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0033] FIG. 1A is a schematic illustration of an exemplary subset of intra prediction modes.

[0034] FIG. 1B is an illustration of exemplary intra prediction directions.

[0035] FIG. 2 shows a current block (201) and surrounding samples in accordance with an embodiment.

[0036] FIG. 3 is a schematic illustration of a simplified block diagram of a communication system (300) in accordance with an embodiment.

[0037] FIG. 4 is a schematic illustration of a simplified block diagram of a communication system (400) in accordance with an embodiment.

[0038] FIG. 5 is a schematic illustration of a simplified block diagram of a decoder in accordance with an embodiment.

[0039] FIG. 6 is a schematic illustration of a simplified block diagram of an encoder in accordance with an embodiment.

[0040] FIG. 7 shows a block diagram of an encoder in accordance with another embodiment.

[0041] FIG. 8 shows a block diagram of a decoder in accordance with another embodiment.

[0042] FIG. 9A shows an example of a block-wise image coding according to an embodiment of the disclosure.

[0043] FIG. 9B shows an exemplary NIC framework according to an embodiment of the disclosure.

[0044] FIG. 10 shows an exemplary convolution neural network (CNN) of a main encoder network according to an embodiment of the disclosure.

[0045] FIG. 11 shows an exemplary CNN of a main decoder network according to an embodiment of the disclosure.

[0046] FIG. 12 shows an exemplary CNN of a hyper encoder according to an embodiment of the disclosure.

[0047] FIG. 13 shows an exemplary CNN of a hyper decoder according to an embodiment of the disclosure.

[0048] FIG. 14 shows an exemplary CNN of a context model network according to an embodiment of the disclosure.

[0049] FIG. 15 shows an exemplary CNN of an entropy parameter network according to an embodiment of the disclosure.

[0050] FIG. 16A shows an exemplary video encoder according to an embodiment of the disclosure.

[0051] FIG. 16B shows an exemplary video decoder according to an embodiment of the disclosure.

[0052] FIG. 17 shows an exemplary video encoder according to an embodiment of the disclosure.

[0053] FIG. 18 shows an exemplary video decoder according to an embodiment of the disclosure.

[0054] FIG. 19A shows an example of block-wise image filtering according to an embodiment of the disclosure.

[0055] FIG. 19B shows an exemplary post-filtering module and a NIC framework according to an embodiment of the disclosure.

[0056] FIG. 20A shows an exemplary video decoder according to an embodiment of the disclosure.

[0057] FIG. 20B shows an exemplary video decoder according to an embodiment of the disclosure.

[0058] FIGs. 21A-21C show an exemplary deblocking process according to an embodiment of the disclosure.

[0059] FIG. 22 shows an example of boundary regions including samples of more than two blocks according to embodiments of the disclosure.

[0060] FIG. 23 shows an exemplary deblocking process based on multiple deblocking models according to an embodiment of the disclosure.

[0061] FIG. 24 shows an exemplary enhancement process according to an embodiment of the disclosure.

[0062] FIG. 25 shows an exemplary enhancement process according to an embodiment of the disclosure.

[0063] FIG. 26 shows an exemplary image-level enhancement process according to an embodiment of the disclosure.

[0064] FIG. 27 shows a flow chart outlining an encoding process according to an embodiment of the disclosure.

[0065] FIG. 28 shows a flow chart outlining a decoding process according to an embodiment of the disclosure.

[0066] FIG. 29 is a schematic illustration of a computer system in accordance with an embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS

[0067] FIG. 3 illustrates a simplified block diagram of a communication system (300) according to an embodiment of the present disclosure. The communication system (300) includes a plurality of terminal devices that can communicate with each other, via, for example, a network (350). For example, the communication system (300) includes a first pair of terminal devices (310) and (320) interconnected via the network (350). In the FIG. 3 example, the first pair of terminal devices (310) and (320) performs unidirectional transmission of data. For example, the terminal device (310) may code video data (e.g., a stream of video pictures that are captured by the terminal device (310)) for transmission to the other terminal device (320) via the network (350). The encoded video data can be transmitted in the form of one or more coded video bitstreams. The terminal device (320) may receive the coded video data from the network (350), decode the coded video data to recover the video pictures and display video pictures according to the recovered video data. Unidirectional data transmission may be common in media serving applications and the like.

[0068] In another example, the communication system (300) includes a second pair of terminal devices (330) and (340) that performs bidirectional transmission of coded video data that may occur, for example, during videoconferencing. For bidirectional transmission of

data, in an example, each terminal device of the terminal devices (330) and (340) may code video data (e.g., a stream of video pictures that are captured by the terminal device) for transmission to the other terminal device of the terminal devices (330) and (340) via the network (350). Each terminal device of the terminal devices (330) and (340) also may receive the coded video data transmitted by the other terminal device of the terminal devices (330) and (340), and may decode the coded video data to recover the video pictures and may display video pictures at an accessible display device according to the recovered video data.

[0069] In the FIG. 3 example, the terminal devices (310), (320), (330) and (340) may be illustrated as servers, personal computers and smart phones but the principles of the present disclosure may be not so limited. Embodiments of the present disclosure find application with laptop computers, tablet computers, media players and/or dedicated video conferencing equipment. The network (350) represents any number of networks that convey coded video data among the terminal devices (310), (320), (330) and (340), including for example wireline (wired) and/or wireless communication networks. The communication network (350) may exchange data in circuit-switched and/or packet-switched channels. Representative networks include telecommunications networks, local area networks, wide area networks and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network (350) may be immaterial to the operation of the present disclosure unless explained herein below.

[0070] FIG. 4 illustrates, as an example for an application for the disclosed subject matter, the placement of a video encoder and a video decoder in a streaming environment. The disclosed subject matter can be equally applicable to other video enabled applications, including, for example, video conferencing, digital TV, storing of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[0071] A streaming system may include a capture subsystem (413), that can include a video source (401), for example a digital camera, creating for example a stream of video pictures (402) that are uncompressed. In an example, the stream of video pictures (402) includes samples that are taken by the digital camera. The stream of video pictures (402), depicted as a bold line to emphasize a high data volume when compared to encoded video data (404) (or coded video bitstreams), can be processed by an electronic device (420) that includes a video encoder (403) coupled to the video source (401). The video encoder (403) can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The encoded video data (404)

(or encoded video bitstream), depicted as a thin line to emphasize the lower data volume when compared to the stream of video pictures (402), can be stored on a streaming server (405) for future use. One or more streaming client subsystems, such as client subsystems (406) and (408) in FIG. 4 can access the streaming server (405) to retrieve copies (407) and (409) of the encoded video data (404). A client subsystem (406) can include a video decoder (410), for example, in an electronic device (430). The video decoder (410) decodes the incoming copy (407) of the encoded video data (404) and creates an outgoing stream of video pictures (411) that can be rendered on a display (412) (e.g., display screen) or other rendering device (not depicted). In some streaming systems, the encoded video data (404), (407), and (409) (e.g., video bitstreams) can be encoded according to certain video coding/compression standards. Examples of those standards include ITU-T Recommendation H.265. In an example, a video coding standard under development is informally known as Versatile Video Coding (VVC). The disclosed subject matter may be used in the context of VVC.

[0072] It is noted that the electronic devices (420) and (430) can include other components (not shown). For example, the electronic device (420) can include a video decoder (not shown) and the electronic device (430) can include a video encoder (not shown) as well.

[0073] FIG. 5 shows a block diagram of a video decoder (510) according to an embodiment of the present disclosure. The video decoder (510) can be included in an electronic device (530). The electronic device (530) can include a receiver (531) (e.g., receiving circuitry). The video decoder (510) can be used in the place of the video decoder (410) in the FIG. 4 example.

[0074] The receiver (531) may receive one or more coded video sequences to be decoded by the video decoder (510); in the same or another embodiment, one coded video sequence at a time, where the decoding of each coded video sequence is independent from other coded video sequences. The coded video sequence may be received from a channel (501), which may be a hardware/software link to a storage device which stores the encoded video data. The receiver (531) may receive the encoded video data with other data, for example, coded audio data and/or ancillary data streams, that may be forwarded to their respective using entities (not depicted). The receiver (531) may separate the coded video sequence from the other data. To combat network jitter, a buffer memory (515) may be coupled in between the receiver (531) and an entropy decoder / parser (520) ("parser (520)" henceforth). In certain applications, the buffer memory (515) is part of the video decoder

(510). In others, it can be outside of the video decoder (510) (not depicted). In still others, there can be a buffer memory (not depicted) outside of the video decoder (510), for example to combat network jitter, and in addition another buffer memory (515) inside the video decoder (510), for example to handle playout timing. When the receiver (531) is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer memory (515) may not be needed, or can be small. For use on best effort packet networks such as the Internet, the buffer memory (515) may be required, can be comparatively large and can be advantageously of adaptive size, and may at least partially be implemented in an operating system or similar elements (not depicted) outside of the video decoder (510).

[0075] The video decoder (510) may include the parser (520) to reconstruct symbols (521) from the coded video sequence. Categories of those symbols include information used to manage operation of the video decoder (510), and potentially information to control a rendering device such as a render device (512) (e.g., a display screen) that is not an integral part of the electronic device (530) but can be coupled to the electronic device (530), as was shown in FIG. 5. The control information for the rendering device(s) may be in the form of Supplemental Enhancement Information (SEI messages) or Video Usability Information (VUI) parameter set fragments (not depicted). The parser (520) may parse / entropy-decode the coded video sequence that is received. The coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow various principles, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser (520) may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameter corresponding to the group. Subgroups can include Groups of Pictures (GOPs), pictures, tiles, slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The parser (520) may also extract from the coded video sequence information such as transform coefficients, quantizer parameter values, motion vectors, and so forth.

[0076] The parser (520) may perform an entropy decoding / parsing operation on the video sequence received from the buffer memory (515), so as to create symbols (521).

[0077] Reconstruction of the symbols (521) can involve multiple different units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. Which units are involved, and how, can be

controlled by the subgroup control information that was parsed from the coded video sequence by the parser (520). The flow of such subgroup control information between the parser (520) and the multiple units below is not depicted for clarity.

[0078] Beyond the functional blocks already mentioned, the video decoder (510) can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these units interact closely with each other and can, at least partly, be integrated into each other. However, for the purpose of describing the disclosed subject matter, the conceptual subdivision into the functional units below is appropriate.

[0079] A first unit is the scaler / inverse transform unit (551). The scaler / inverse transform unit (551) receives a quantized transform coefficient as well as control information, including which transform to use, block size, quantization factor, quantization scaling matrices, etc. as symbol(s) (521) from the parser (520). The scaler / inverse transform unit (551) can output blocks comprising sample values, that can be input into aggregator (555).

[0080] In some cases, the output samples of the scaler / inverse transform (551) can pertain to an intra coded block; that is: a block that is not using predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit (552). In some cases, the intra picture prediction unit (552) generates a block of the same size and shape of the block under reconstruction, using surrounding already reconstructed information fetched from the current picture buffer (558). The current picture buffer (558) buffers, for example, partly reconstructed current picture and/or fully reconstructed current picture. The aggregator (555), in some cases, adds, on a per sample basis, the prediction information the intra prediction unit (552) has generated to the output sample information as provided by the scaler / inverse transform unit (551).

[0081] In other cases, the output samples of the scaler / inverse transform unit (551) can pertain to an inter coded, and potentially motion compensated block. In such a case, a motion compensation prediction unit (553) can access reference picture memory (557) to fetch samples used for prediction. After motion compensating the fetched samples in accordance with the symbols (521) pertaining to the block, these samples can be added by the aggregator (555) to the output of the scaler / inverse transform unit (551) (in this case called the residual samples or residual signal) so as to generate output sample information. The addresses within the reference picture memory (557) from where the motion compensation

prediction unit (553) fetches prediction samples can be controlled by motion vectors, available to the motion compensation prediction unit (553) in the form of symbols (521) that can have, for example X, Y, and reference picture components. Motion compensation also can include interpolation of sample values as fetched from the reference picture memory (557) when sub-sample exact motion vectors are in use, motion vector prediction mechanisms, and so forth.

[0082] The output samples of the aggregator (555) can be subject to various loop filtering techniques in the loop filter unit (556). Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video sequence (also referred to as coded video bitstream) and made available to the loop filter unit (556) as symbols (521) from the parser (520), but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered sample values.

[0083] The output of the loop filter unit (556) can be a sample stream that can be output to the render device (512) as well as stored in the reference picture memory (557) for use in future inter-picture prediction.

[0084] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future prediction. For example, once a coded picture corresponding to a current picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, the parser (520)), the current picture buffer (558) can become a part of the reference picture memory (557), and a fresh current picture buffer can be reallocated before commencing the reconstruction of the following coded picture.

[0085] The video decoder (510) may perform decoding operations according to a predetermined video compression technology in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that the coded video sequence adheres to both the syntax of the video compression technology or standard and the profiles as documented in the video compression technology or standard. Specifically, a profile can select certain tools as the only tools available for use under that profile from all the tools available in the video compression technology or standard. Also necessary for compliance can be that the complexity of the coded video sequence is within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size,

maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[0086] In an embodiment, the receiver (531) may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder (510) to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[0087] FIG. 6 shows a block diagram of a video encoder (603) according to an embodiment of the present disclosure. The video encoder (603) is included in an electronic device (620). The electronic device (620) includes a transmitter (640) (e.g., transmitting circuitry). The video encoder (603) can be used in the place of the video encoder (403) in the FIG. 4 example.

[0088] The video encoder (603) may receive video samples from a video source (601) (that is not part of the electronic device (620) in the FIG. 6 example) that may capture video image(s) to be coded by the video encoder (603). In another example, the video source (601) is a part of the electronic device (620).

[0089] The video source (601) may provide the source video sequence to be coded by the video encoder (603) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, …), any colorspace (for example, BT.601 Y CrCB, RGB, …), and any suitable sampling structure (for example Y CrCb 4:2:0, Y CrCb 4:4:4). In a media serving system, the video source (601) may be a storage device storing previously prepared video. In a videoconferencing system, the video source (601) may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual pictures that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, etc. in use. A person skilled in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[0090] According to an embodiment, the video encoder (603) may code and compress the pictures of the source video sequence into a coded video sequence (643) in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed is one function of a controller (650). In some embodiments, the controller (650) controls other functional units as described below and is functionally coupled to the other functional units. The coupling is not depicted for clarity. Parameters set by the controller (650) can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, …), picture size, group of pictures (GOP) layout, maximum motion vector search range, and so forth. The controller (650) can be configured to have other suitable functions that pertain to the video encoder (603) optimized for a certain system design.

[0091] In some embodiments, the video encoder (603) is configured to operate in a coding loop. As an oversimplified description, in an example, the coding loop can include a source coder (630) (e.g., responsible for creating symbols, such as a symbol stream, based on an input picture to be coded, and a reference picture(s)), and a (local) decoder (633) embedded in the video encoder (603). The decoder (633) reconstructs the symbols to create the sample data in a similar manner as a (remote) decoder also would create (as any compression between symbols and coded video bitstream is lossless in the video compression technologies considered in the disclosed subject matter). The reconstructed sample stream (sample data) is input to the reference picture memory (634). As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the content in the reference picture memory (634) is also bit exact between the local encoder and remote encoder. In other words, the prediction part of an encoder "sees" as reference picture samples exactly the same sample values as a decoder would "see" when using prediction during decoding. This fundamental principle of reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is used in some related arts as well.

[0092] The operation of the "local" decoder (633) can be the same as of a "remote" decoder, such as the video decoder (510), which has already been described in detail above in conjunction with FIG. 5. Briefly referring also to FIG. 5, however, as symbols are available and encoding/decoding of symbols to a coded video sequence by an entropy coder (645) and the parser (520) can be lossless, the entropy decoding parts of the video decoder (510),

including the buffer memory (515), and parser (520) may not be fully implemented in the local decoder (633).

[0093] In an embodiment, a decoder technology except the parsing/entropy decoding that is present in a decoder is present, in an identical or a substantially identical functional form, in a corresponding encoder. Accordingly, the disclosed subject matter focuses on decoder operation. The description of encoder technologies can be abbreviated as they are the inverse of the comprehensively described decoder technologies. In certain areas a more detail description is provided below.

[0094] During operation, in some examples, the source coder (630) may perform motion compensated predictive coding, which codes an input picture predictively with reference to one or more previously coded picture from the video sequence that were designated as "reference pictures." In this manner, the coding engine (632) codes differences between pixel blocks of an input picture and pixel blocks of reference picture(s) that may be selected as prediction reference(s) to the input picture.

[0095] The local video decoder (633) may decode coded video data of pictures that may be designated as reference pictures, based on symbols created by the source coder (630). Operations of the coding engine (632) may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in FIG. 6), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder (633) replicates decoding processes that may be performed by the video decoder on reference pictures and may cause reconstructed reference pictures to be stored in the reference picture memory (634). In this manner, the video encoder (603) may store copies of reconstructed reference pictures locally that have common content as the reconstructed reference pictures that will be obtained by a far-end video decoder (absent transmission errors).

[0096] The predictor (635) may perform prediction searches for the coding engine (632). That is, for a new picture to be coded, the predictor (635) may search the reference picture memory (634) for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor (635) may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by search results obtained by the predictor (635), an input picture may have

prediction references drawn from multiple reference pictures stored in the reference picture memory (634).

[0097] The controller (650) may manage coding operations of the source coder (630), including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0098] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder (645). The entropy coder (645) translates the symbols as generated by the various functional units into a coded video sequence, by lossless compressing the symbols according to technologies such as Huffman coding, variable length coding, arithmetic coding, and so forth.

[0099] The transmitter (640) may buffer the coded video sequence(s) as created by the entropy coder (645) to prepare for transmission via a communication channel (660), which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter (640) may merge coded video data from the video encoder (603) with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[0100] The controller (650) may manage operation of the video encoder (603). During coding, the controller (650) may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following picture types:

[0101] An Intra Picture (I picture) may be one that may be coded and decoded without using any other picture in the sequence as a source of prediction. Some video codecs allow for different types of intra pictures, including, for example Independent Decoder Refresh ("IDR") Pictures. A person skilled in the art is aware of those variants of I pictures and their respective applications and features.

[0102] A predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[0103] A bi-directionally predictive picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can use more than two reference pictures and associated metadata for the reconstruction of a single block.

[0104] Source pictures commonly may be subdivided spatially into a plurality of sample blocks (for example, blocks of 4x4, 8x8, 4x8, or 16x16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks' respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference picture. Blocks of B pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures.

[0105] The video encoder (603) may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video encoder (603) may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data, therefore, may conform to a syntax specified by the video coding technology or standard being used.

[0106] In an embodiment, the transmitter (640) may transmit additional data with the encoded video. The source coder (630) may include such data as part of the coded video sequence. Additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, SEI messages, VUI parameter set fragments, and so on.

[0107] A video may be captured as a plurality of source pictures (video pictures) in a temporal sequence. Intra-picture prediction (often abbreviated to intra prediction) makes use of spatial correlation in a given picture, and inter-picture prediction makes uses of the (temporal or other) correlation between the pictures. In an example, a specific picture under encoding/decoding, which is referred to as a current picture, is partitioned into blocks. When a block in the current picture is similar to a reference block in a previously coded and still buffered reference picture in the video, the block in the current picture can be coded by a vector that is referred to as a motion vector. The motion vector points to the reference block in the reference picture, and can have a third dimension identifying the reference picture, in case multiple reference pictures are in use.

[0108] In some embodiments, a bi-prediction technique can be used in the inter-picture prediction. According to the bi-prediction technique, two reference pictures, such as

a first reference picture and a second reference picture that are both prior in decoding order to the current picture in the video (but may be in the past and future, respectively, in display order) are used. A block in the current picture can be coded by a first motion vector that points to a first reference block in the first reference picture, and a second motion vector that points to a second reference block in the second reference picture. The block can be predicted by a combination of the first reference block and the second reference block.

[0109] Further, a merge mode technique can be used in the inter-picture prediction to improve coding efficiency.

[0110] According to some embodiments of the disclosure, predictions, such as inter-picture predictions and intra-picture predictions are performed in the unit of blocks. For example, according to the HEVC standard, a picture in a sequence of video pictures is partitioned into coding tree units (CTU) for compression, the CTUs in a picture have the same size, such as 64x64 pixels, 32x32 pixels, or 16x16 pixels. In general, a CTU includes three coding tree blocks (CTBs), which are one luma CTB and two chroma CTBs. Each CTU can be recursively quadtree split into one or multiple coding units (CUs). For example, a CTU of 64x64 pixels can be split into one CU of 64x64 pixels, or 4 CUs of 32x32 pixels, or 16 CUs of 16x16 pixels. In an example, each CU is analyzed to determine a prediction type for the CU, such as an inter prediction type or an intra prediction type. The CU is split into one or more prediction units (PUs) depending on the temporal and/or spatial predictability. Generally, each PU includes a luma prediction block (PB), and two chroma PBs. In an embodiment, a prediction operation in coding (encoding/decoding) is performed in the unit of a prediction block. Using a luma prediction block as an example of a prediction block, the prediction block includes a matrix of values (e.g., luma values) for pixels, such as 8x8 pixels, 16x16 pixels, 8x16 pixels, 16x8 pixels, and the like.

[0111] FIG. 7 shows a diagram of a video encoder (703) according to another embodiment of the disclosure. The video encoder (703) is configured to receive a processing block (e.g., a prediction block) of sample values within a current video picture in a sequence of video pictures, and encode the processing block into a coded picture that is part of a coded video sequence. In an example, the video encoder (703) is used in the place of the video encoder (403) in the FIG. 4 example.

[0112] In an HEVC example, the video encoder (703) receives a matrix of sample values for a processing block, such as a prediction block of 8x8 samples, and the like. The video encoder (703) determines whether the processing block is best coded using intra mode,

inter mode, or bi-prediction mode using, for example, rate-distortion optimization. When the processing block is to be coded in intra mode, the video encoder (703) may use an intra prediction technique to encode the processing block into the coded picture; and when the processing block is to be coded in inter mode or bi-prediction mode, the video encoder (703) may use an inter prediction or bi-prediction technique, respectively, to encode the processing block into the coded picture. In certain video coding technologies, merge mode can be an inter picture prediction submode where the motion vector is derived from one or more motion vector predictors without the benefit of a coded motion vector component outside the predictors. In certain other video coding technologies, a motion vector component applicable to the subject block may be present. In an example, the video encoder (703) includes other components, such as a mode decision module (not shown) to determine the mode of the processing blocks.

[0113] In the FIG. 7 example, the video encoder (703) includes the inter encoder (730), an intra encoder (722), a residue calculator (723), a switch (726), a residue encoder (724), a general controller (721), and an entropy encoder (725) coupled together as shown in FIG. 7.

[0114] The inter encoder (730) is configured to receive the samples of the current block (e.g., a processing block), compare the block to one or more reference blocks in reference pictures (e.g., blocks in previous pictures and later pictures), generate inter prediction information (e.g., description of redundant information according to inter encoding technique, motion vectors, merge mode information), and calculate inter prediction results (e.g., predicted block) based on the inter prediction information using any suitable technique. In some examples, the reference pictures are decoded reference pictures that are decoded based on the encoded video information.

[0115] The intra encoder (722) is configured to receive the samples of the current block (e.g., a processing block), in some cases compare the block to blocks already coded in the same picture, generate quantized coefficients after transform, and in some cases also intra prediction information (e.g., an intra prediction direction information according to one or more intra encoding techniques). In an example, the intra encoder (722) also calculates intra prediction results (e.g., predicted block) based on the intra prediction information and reference blocks in the same picture.

[0116] The general controller (721) is configured to determine general control data and control other components of the video encoder (703) based on the general control data.

In an example, the general controller (721) determines the mode of the block, and provides a control signal to the switch (726) based on the mode. For example, when the mode is the intra mode, the general controller (721) controls the switch (726) to select the intra mode result for use by the residue calculator (723), and controls the entropy encoder (725) to select the intra prediction information and include the intra prediction information in the bitstream; and when the mode is the inter mode, the general controller (721) controls the switch (726) to select the inter prediction result for use by the residue calculator (723), and controls the entropy encoder (725) to select the inter prediction information and include the inter prediction information in the bitstream.

[0117] The residue calculator (723) is configured to calculate a difference (residue data) between the received block and prediction results selected from the intra encoder (722) or the inter encoder (730). The residue encoder (724) is configured to operate based on the residue data to encode the residue data to generate the transform coefficients. In an example, the residue encoder (724) is configured to convert the residue data from a spatial domain to a frequency domain, and generate the transform coefficients. The transform coefficients are then subject to quantization processing to obtain quantized transform coefficients. In various embodiments, the video encoder (703) also includes a residue decoder (728). The residue decoder (728) is configured to perform inverse-transform, and generate the decoded residue data. The decoded residue data can be suitably used by the intra encoder (722) and the inter encoder (730). For example, the inter encoder (730) can generate decoded blocks based on the decoded residue data and inter prediction information, and the intra encoder (722) can generate decoded blocks based on the decoded residue data and the intra prediction information. The decoded blocks are suitably processed to generate decoded pictures and the decoded pictures can be buffered in a memory circuit (not shown) and used as reference pictures in some examples.

[0118] The entropy encoder (725) is configured to format the bitstream to include the encoded block. The entropy encoder (725) is configured to include various information according to a suitable standard, such as the HEVC standard. In an example, the entropy encoder (725) is configured to include the general control data, the selected prediction information (e.g., intra prediction information or inter prediction information), the residue information, and other suitable information in the bitstream. Note that, according to the disclosed subject matter, when coding a block in the merge submode of either inter mode or bi-prediction mode, there is no residue information.

[0119] FIG. 8 shows a diagram of a video decoder (810) according to another embodiment of the disclosure. The video decoder (810) is configured to receive coded pictures that are part of a coded video sequence, and decode the coded pictures to generate reconstructed pictures. In an example, the video decoder (810) is used in the place of the video decoder (410) in the FIG. 4 example.

[0120] In the FIG. 8 example, the video decoder (810) includes an entropy decoder (871), an inter decoder (880), a residue decoder (873), a reconstruction module (874), and an intra decoder (872) coupled together as shown in FIG. 8.

[0121] The entropy decoder (871) can be configured to reconstruct, from the coded picture, certain symbols that represent the syntax elements of which the coded picture is made up. Such symbols can include, for example, the mode in which a block is coded (such as, for example, intra mode, inter mode, bi-predicted mode, the latter two in merge submode or another submode), prediction information (such as, for example, intra prediction information or inter prediction information) that can identify certain sample or metadata that is used for prediction by the intra decoder (872) or the inter decoder (880), respectively, residual information in the form of, for example, quantized transform coefficients, and the like. In an example, when the prediction mode is inter or bi-predicted mode, the inter prediction information is provided to the inter decoder (880); and when the prediction type is the intra prediction type, the intra prediction information is provided to the intra decoder (872). The residual information can be subject to inverse quantization and is provided to the residue decoder (873).

[0122] The inter decoder (880) is configured to receive the inter prediction information, and generate inter prediction results based on the inter prediction information.

[0123] The intra decoder (872) is configured to receive the intra prediction information, and generate prediction results based on the intra prediction information.

[0124] The residue decoder (873) is configured to perform inverse quantization to extract de-quantized transform coefficients, and process the de-quantized transform coefficients to convert the residual from the frequency domain to the spatial domain. The residue decoder (873) may also require certain control information (to include the Quantizer Parameter (QP)), and that information may be provided by the entropy decoder (871) (data path not depicted as this may be low volume control information only).

[0125] The reconstruction module (874) is configured to combine, in the spatial domain, the residual as output by the residue decoder (873) and the prediction results (as

output by the inter or intra prediction modules as the case may be) to form a reconstructed block, that may be part of the reconstructed picture, which in turn may be part of the reconstructed video. It is noted that other suitable operations, such as a deblocking operation and the like, can be performed to improve the visual quality.

[0126] It is noted that the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using any suitable technique. In an embodiment, the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using one or more integrated circuits. In another embodiment, the video encoders (403), (603), and (603), and the video decoders (410), (510), and (810) can be implemented using one or more processors that execute software instructions.

[0127] This disclosure describes video coding technologies related to neural image compression technologies and/or neural video compression technologies, such as artificial intelligence (AI) based neural image compression (NIC). Aspects of the disclosure include content-adaptive online training in NIC, such as content-adaptive online training NIC methods with post filtering for an end-to-end (E2E) optimized image coding framework based on neural networks. A neural network (NN) can include an artificial neural network (ANN), such as a deep neural network (DNN), a convolution neural network (CNN), or the like.

[0128] In an embodiment, a related hybrid video codec is difficult to be optimized as a whole. For example, an improvement of a single module (e.g., an encoder) in the hybrid video codec may not result in a coding gain in the overall performance. In a NN-based video coding framework, different modules can be jointly optimized from an input to an output to improve a final objective (e.g., rate-distortion performance, such as a rate-distortion loss L described in the disclosure) by performing a learning process or a training process (e.g., a machine learning process), and thus resulting in an E2E optimized NIC.

[0129] An exemplary NIC framework or system can be described as follows. The NIC framework can use an input block x as an input to a neural network encoder (e.g., an encoder based on neural networks such as DNNs) to compute a compressed representation (e.g., a compact representation) $\hat{x}$ that can be compact, for example, for storage and transmission purposes. A neural network decoder (e.g., a decoder based on neural networks such as DNNs) can use the compressed representation $\hat{x}$ as an input to reconstruct an output block (also referred to as a reconstructed block) $\bar{x}$. In various embodiments, the input block x and reconstructed block $\bar{x}$ are in a spatial domain and the compressed representation $\hat{x}$ is in a

domain different from the spatial domain. In some examples, the compressed representation $\hat{x}$ is quantized and entropy coded.

[0130] In some examples, a NIC framework can use a variational autoencoder (VAE) structure. In the VAE structure, the neural network encoder can directly use the entire input block x as the input to the neural network encoder. The entire input block x can pass through a set of neural network layers that work as a black box to compute the compressed representation $\hat{x}$. The compressed representation $\hat{x}$ is an output of the neural network encoder. The neural network decoder can take the entire compressed representation $\hat{x}$ as an input. The compressed representation $\hat{x}$ can pass through another set of neural network layers that work as another black box to compute the reconstructed block $\bar{x}$. A rate-distortion (R-D) loss L $(x, \bar{x}, \hat{x})$ can be optimized to achieve a trade-off between a distortion loss $D(x, \bar{x})$ of the reconstructed block $\bar{x}$ and bit consumption R of the compact representation $\hat{x}$ with a trade-off hyperparameter λ.

$$\mathrm{L}\ (x, \bar{x}, \hat{x}) = \lambda\, D(x, \bar{x}) + \mathrm{R}\ (\hat{x}) \qquad \text{Eq. 1}$$

[0131] A neural network (e.g., an ANN) can learn to perform tasks from examples, without task-specific programming. An ANN can be configured with connected nodes or artificial neurons. A connection between nodes can transmit a signal from a first node to a second node (e.g., a receiving node), and the signal can be modified by a weight which can be indicated by a weight coefficient for the connection. The receiving node can process signal(s) (i.e., input signal(s) for the receiving node) from node(s) that transmit the signal(s) to the receiving node and then generate an output signal by applying a function to the input signals. The function can be a linear function. In an example, the output signal is a weighted summation of the input signal(s). In an example, the output signal is further modified by a bias which can be indicated by a bias term, and thus the output signal is a sum of the bias and the weighted summation of the input signal(s). The function can include a nonlinear operation, for example, on the weighted sum or the sum of the bias and the weighted summation of the input signal(s). The output signal can be sent to node(s) (downstream node(s)) connected to the receiving node). The ANN can be represented or configured by parameters (e.g., weights of the connections and/or biases). The weights and/or the biases can be obtained by training the ANN with examples where the weights and/or the biases can be iteratively adjusted. The trained ANN configured with the determined weights and/or the determined biases can be used to perform tasks.

[0132] Nodes in an ANN can be organized in any suitable architecture. In various embodiments, nodes in an ANN are organized in layers including an input layer that receives input signal(s) to the ANN and an output layer that outputs output signal(s) from the ANN. In an embodiment, the ANN further includes layer(s) such as hidden layer(s) between the input layer and the output layer. Different layers may perform different kinds of transformations on respective inputs of the different layers. Signals can travel from the input layer to the output layer.

[0133] An ANN with multiple layers between an input layer and an output layer can be referred to as a DNN. In an embodiment, a DNN is a feedforward network where data flows from the input layer to the output layer without looping back. In an example, a DNN is a fully connected network where each node in one layer is connected to all nodes in the next layer. In an embodiment, a DNN is a recurrent neural network (RNN) where data can flow in any direction. In an embodiment, a DNN is a CNN.

[0134] A CNN can include an input layer, an output layer, and hidden layer(s) between the input layer and the output layer. The hidden layer(s) can include convolutional layer(s) (e.g., used in an encoder) that perform convolutions, such as a two-dimensional (2D) convolution. In an embodiment, a 2D convolution performed in a convolution layer is between a convolution kernel (also referred to as a filter or a channel, such as a 5 x 5 matrix) and an input signal (e.g., a 2D matrix such as a 2D block, a 256 x 256 matrix) to the convolution layer. In various examples, a dimension of the convolution kernel (e.g., 5 x 5) is smaller than a dimension of the input signal (e.g., 256 x 256). Thus, a portion (e.g., a 5 x 5 area) in the input signal (e.g., a 256 x 256 matrix) that is covered by the convolution kernel is smaller than an area (e.g., a 256 x 256 area) of the input signal, and thus can be referred to as a receptive field in the respective node in the next layer.

[0135] During the convolution, a dot product of the convolution kernel and the corresponding receptive field in the input signal is calculated. Thus, each element of the convolution kernel is a weight that is applied to a corresponding sample in the receptive field, and thus the convolution kernel includes weights. For example, a convolution kernel represented by a 5 x 5 matrix has 25 weights. In some examples, a bias is applied to the output signal of the convolution layer, and the output signal is based on a sum of the dot product and the bias.

[0136] The convolution kernel can shift along the input signal (e.g., a 2D matrix) by a size referred to as a stride, and thus the convolution operation generates a feature map or an

activation map (e.g., another 2D matrix), which in turn contributes to an input of the next layer in the CNN. For example, the input signal is a 2D block having 256 x 256 samples, a stride is 2 samples (e.g., a stride of 2). For the stride of 2, the convolution kernel shifts along an X direction (e.g., a horizontal direction) and/or a Y direction (e.g., a vertical direction) by 2 samples.

[0137] Multiple convolution kernels can be applied in the same convolution layer to the input signal to generate multiple feature maps, respectively, where each feature map can represent a specific feature of the input signal. In general, a convolution layer with N channels (i.e., N convolution kernels), each convolution kernel having M x M samples, and a stride S can be specified as Conv: MxM cN sS. For example, a convolution layer with 192 channels, each convolution kernel having 5 x 5 samples, and a stride of 2 is specified as Conv: 5x5 c192 s2. The hidden layer(s) can include deconvolutional layer(s) (e.g., used in a decoder) that perform deconvolutions, such as a 2D deconvolution. A deconvolution is an inverse of a convolution. A deconvolution layer with 192 channels, each deconvolution kernel having 5 x 5 samples, and a stride of 2 is specified as DeConv: 5x5 c192 s2.

[0138] In various embodiments, a CNN has the following benefits. A number of learnable parameters (i.e., parameters to be trained) in a CNN can be significantly smaller than a number of learnable parameters in a DNN, such as a feedforward DNN. In the CNN, a relatively large number of nodes can share a same filter (e.g., same weights) and a same bias (if the bias is used), and thus the memory footprint can be reduced because a single bias and a single vector of weights can be used across all receptive fields that share the same filter. For example, for an input signal having 100 x 100 samples, a convolution layer with a convolution kernel having 5 x 5 samples has 25 learnable parameters (e.g., weights). If a bias is used, then one channel uses 26 learnable parameters (e.g., 25 weights and one bias). If the convolution layer has N channels, the total learnable parameters is 26 x N. On the other hand, for a fully connected a layer in a DNN, 100 x 100 (i.e., 10000) weights are used for each node in the next layer. If the next layer has L nodes, then the total learnable parameters is 10000 x L.

[0139] A CNN can further include one or more other layer(s), such as pooling layer(s), fully connected layer(s) that can connect every node in one layer to every node in another layer, normalization layer(s), and/or the like. Layers in a CNN can be arranged in any suitable order and in any suitable architecture (e.g., a feed-forward architecture, a recurrent

architecture). In an example, a convolutional layer is followed by other layer(s), such as pooling layer(s), fully connected layer(s), normalization layer(s), and/or the like.

[0140] A pooling layer can be used to reduce dimensions of data by combining outputs from a plurality of nodes at one layer into a single node in the next layer. A pooling operation for a pooling layer having a feature map as an input is described below. The description can be suitably adapted to other input signals. The feature map can be divided into sub-regions (e.g., rectangular sub-regions), and features in the respective sub-regions can be independently down-sampled (or pooled) to a single value, for example, by taking an average value in an average pooling or a maximum value in a max pooling.

[0141] The pooling layer can perform a pooling, such as a local pooling, a global pooling, a max pooling, an average pooling, and/or the like. A pooling is a form of nonlinear down-sampling. A local pooling combines a small number of nodes (e.g., a local cluster of nodes, such as 2 x 2 nodes) in the feature map. A global pooling can combine all nodes, for example, of the feature map.

[0142] The pooling layer can reduce a size of the representation, and thus reduce a number of parameters, a memory footprint, and an amount of computation in a CNN. In an example, a pooling layer is inserted between successive convolutional layers in a CNN. In an example, a pooling layer is followed by an activation function, such as a rectified linear unit (ReLU) layer. In an example, a pooling layer is omitted between successive convolutional layers in a CNN.

[0143] A normalization layer can be an ReLU, a leaky ReLU, a generalized divisive normalization (GDN), an inverse GDN (IGDN), or the like. An ReLU can apply a non-saturating activation function to remove negative values from an input signal, such as a feature map, by setting the negative values to zero. A leaky ReLU can have a small slope (e.g., 0.01) for negative values instead of a flat slope (e.g., 0). Accordingly, if a value x is larger than 0, then an output from the leaky ReLU is x. Otherwise, the output from the leaky ReLU is the value x multiplied by the small slope (e.g., 0.01). In an example, the slope is determined before training, and thus is not learnt during training.

[0144] In NN-based image compression methods, such as DNN-based or CNN-based image compression methods, instead of directly encoding an entire image, the block-based or block-wise coding mechanism can be effective for compressing images in a DNN-based video coding standards such as FVC. An entire image can be partitioned into blocks of the same (or various) sizes, and the blocks can be compressed individually. In an embodiment, an

image may be split into blocks with an equal size or non-equal sizes. The spilt blocks instead of the image can be compressed. FIG. 9A shows an example of a block-wise image coding according to an embodiment of the disclosure. An image (980) can be partitioned into blocks, e.g., blocks (981)-(996). The blocks (981)-(996) can be compressed, for example, according to a scanning order. IN an example shown in FIG. 9A, the blocks (981)-(989) are already compressed, and the blocks (990)-(996) are to be compressed.

[0145] An image can be treated as a block. In an embodiment, the image is compressed without being split into blocks. The entire image can be the input of an E2E NIC framework.

[0146] In the descriptions below, for purposes of brevity, a signal is used to refer to an image or a block. Thus, an input signal can refer to an input image or an input block, an encoded signal can refer to an encoded image or an encoded block, and a reconstructed signal can refer to a reconstructed image or a reconstructed block.

[0147] FIG. 9B shows an exemplary NIC framework (900) (e.g., a NIC system) according to an embodiment of the disclosure. The NIC framework (900) can be based on neural networks, such as DNNs and/or CNNs. The NIC framework (900) can be used to compress (e.g., encode) signals (e.g., blocks and/or images) and decompress (e.g., decode or reconstruct) compressed signals (e.g., blocks and/or images) (e.g., encoded blocks or images). The NIC framework (900) can include two sub-neural networks, a first sub-NN (951) and a second sub-NN (952) that are implemented using neural networks.

[0148] The first sub-NN (951) can resemble an autoencoder and can be trained to generate a compressed signal (e.g., a compressed block or a compressed image) $\hat{x}$ of an input signal (e.g., an input block or an input image) x and decompress the compressed signal $\hat{x}$ to obtain a reconstructed signal (e.g., a reconstructed block or a reconstructed image) $\bar{x}$. The first sub-NN (951) can include a plurality of components (or modules), such as a main encoder neural network (or a main encoder network) (911), a quantizer (912), an entropy encoder (913), an entropy decoder (914), and a main decoder neural network (or a main encoder network) (915). Referring to FIG. 9B, the main encoder network (911) can generate a latent or a latent representation y from the input signal x (e.g., a signal to be compressed or encoded). In an example, the main encoder network (911) is implemented using a CNN. A relationship between the latent representation y and the input signal x can be described using Eq. 2.

$$y = f_1(x;\ \theta_1) \qquad \text{Eq. 2}$$

where a parameter θ_1 represents parameters, such as weights used in convolution kernels in the main encoder network (911) and biases (if biases are used in the main encoder network (911)).

[0149] The latent representation y can be quantized using the quantizer (912) to generate a quantized latent $\hat{y}$. The quantized latent $\hat{y}$ can be compressed, for example, using lossless compression by the entropy encoder (913) to generate the compressed signal (e.g., an encoded signal) $\hat{x}$ (931) that is a compressed representation $\hat{x}$ of the input signal x. The entropy encoder (913) can use entropy coding techniques such as Huffman coding, arithmetic coding, or the like. In an example, the entropy encoder (913) uses arithmetic encoding and is an arithmetic encoder. In an example, the encoded signal (931) is transmitted in a coded bitstream.

[0150] The encoded signal (931) can be decompressed (e.g., entropy decoded) by the entropy decoder (914) to generate an output. The entropy decoder (914) can use entropy coding techniques such as Huffman coding, arithmetic coding, or the like that correspond to the entropy encoding techniques used in the entropy encoder (913). In an example, the entropy decoder (914) uses arithmetic decoding and is an arithmetic decoder. In an example, lossless compression is used in the entropy encoder (913), lossless decompression is used in the entropy decoder (914), and noises, such as due to the transmission of the encoded signal (931) are omissible, the output from the entropy decoder (914) is the quantized latent $\hat{y}$.

[0151] The main decoder network (915) can decode the quantized latent $\hat{y}$ to generate the reconstructed signal $\bar{x}$. In an example, the main decoder network (915) is implemented using a CNN. A relationship between the reconstructed signal $\bar{x}$ (i.e., the output of the main decoder network (915)) and the quantized latent $\hat{y}$ (i.e., the input of the main decoder network (915)) can be described using Eq. 3.

$$\bar{x} = f_2(\hat{y};\ \theta_2) \qquad \text{Eq. 3}$$

where a parameter θ_2 represents parameters, such as weights used in convolution kernels in the main decoder network (915) and biases (if biases are used in the main decoder network (915)). Thus, the first sub-NN (951) can compress (e.g., encode) the input signal x to obtain the encoded signal (931) and decompress (e.g., decode) the encoded signal (931) to obtain the reconstructed signal $\bar{x}$. The reconstructed signal $\bar{x}$ can be different from the input signal x due to quantization loss introduced by the quantizer (912).

[0152] The second sub-NN (952) can learn the entropy model (e.g., a prior probabilistic model) over the quantized latent $\hat{y}$ used for entropy coding. Thus, the entropy

model can be a conditioned entropy model, e.g., a Gaussian mixture model (GMM), a Gaussian scale model (GSM) that is dependent on the input block x. The second sub-NN (952) can include a context model NN (916), an entropy parameter NN (917), a hyper encoder (921), a quantizer (922), an entropy encoder (923), an entropy decoder (924), and a hyper decoder (925). The entropy model used in the context model NN (916) can be an autoregressive model over latent (e.g., the quantized latent $\hat{y}$). In an example, the hyper encoder (921), the quantizer (922), the entropy encoder (923), the entropy decoder (924), and the hyper decoder (925) form a hyper neural network (e.g., a hyperprior NN). The hyper neural network can represent information useful for correcting context-based predictions. Data from the context model NN (916) and the hyper neural network can be combined by the entropy parameter NN (917). The entropy parameter NN (917) can generate parameters, such as mean and scale parameters for the entropy model such as a conditional Gaussian entropy model (e.g., the GMM).

[0153] Referring to FIG. 9B, at an encoder side, the quantized latent $\hat{y}$ from the quantizer (912) is fed into the context model NN (916). At a decoder side, the quantized latent $\hat{y}$ from the entropy decoder (914) is fed into the context model NN (916). The context model NN (916) can be implemented using a neural network, such as a CNN. The context model NN (916) can generate an output $o_{cm,i}$ based on a context $\hat{y}_{<i}$ that is the quantized latent $\hat{y}$ available to the context model NN (916). The context $\hat{y}_{<i}$ can include previously quantized latent at the encoder side or previously entropy decoded quantized latent at the decoder side. A relationship between the output $o_{cm,i}$ and the input (e.g., $\hat{y}_{<i}$) of the context model NN (916) can be described using Eq. 4.

$$o_{cm,i} = f_3(\hat{y}_{<i}\,;\ \theta_3) \qquad \text{Eq. 4}$$

where a parameter θ_3 represents parameters, such as weights used in convolution kernels in the context model NN (916) and biases (if biases are used in the context model NN (916)).

[0154] The output $o_{cm,i}$ from the context model NN (916) and an output o_{hc} from the hyper decoder (925) are fed into the entropy parameter NN (917) to generate an output o_{ep}. The entropy parameter NN (917) can be implemented using a neural network, such as a CNN. A relationship between the output o_{ep} and the inputs (e.g., $o_{cm,i}$ and o_{hc}) of the entropy parameter NN (917) can be described using Eq. 5.

$$o_{ep} = f_4\left(o_{cm,i}, o_{hc};\ \theta_4\right) \qquad \text{Eq. 5}$$

where a parameter θ_4 represents parameters, such as weights used in convolution kernels in the entropy parameter NN (917) and biases (if biases are used in the entropy parameter NN (917)). The output o_{ep} of the entropy parameter NN (917) can be used in determining (e.g., conditioning) the entropy model, and thus the conditioned entropy model can be dependent on the input signal x, for example, via the output o_{hc} from the hyper decoder (925). In an example, the output o_{ep} includes parameters, such as the mean and scale parameters, used to condition the entropy model (e.g., GMM). Referring to FIG. 9B, the entropy model (e.g., the conditioned entropy model) can be employed by the entropy encoder (913) and the entropy decoder (914) in entropy coding and entropy decoding, respectively.

[0155] The second sub-NN (952) can be described below. The latent y can be fed into the hyper encoder (921) to generate a hyper latent z. In an example, the hyper encoder (921) is implemented using a neural network, such as a CNN. A relationship between the hyper latent z and the latent y can be described using Eq. 6.

$$z = f_5(y;\ \theta_5) \qquad \text{Eq. 6}$$

where a parameter θ_5 represents parameters, such as weights used in convolution kernels in the hyper encoder (921) and biases (if biases are used in the hyper encoder (921)).

[0156] The hyper latent z is quantized by the quantizer (922) to generate a quantized latent $\hat{z}$. The quantized latent $\hat{z}$ can be compressed, for example, using lossless compression by the entropy encoder (923) to generate side information, such as encoded bits (932) from the hyper neural network. The entropy encoder (923) can use entropy coding techniques such as Huffman coding, arithmetic coding, or the like. In an example, the entropy encoder (923) uses arithmetic encoding and is an arithmetic encoder. In an example, the side information, such as the encoded bits (932), can be transmitted in the coded bitstream, for example, together with the encoded signal (931).

[0157] The side information, such as the encoded bits (932), can be decompressed (e.g., entropy decoded) by the entropy decoder (924) to generate an output. The entropy decoder (924) can use entropy coding techniques such as Huffman coding, arithmetic coding, or the like. In an example, the entropy decoder (924) uses arithmetic decoding and is an arithmetic decoder. In an example, lossless compression is used in the entropy encoder (923), lossless decompression is used in the entropy decoder (924), and noises, such as due to the transmission of the side information are omissible, the output from the entropy decoder (924) can be the quantized latent $\hat{z}$. The hyper decoder (925) can decode the quantized latent $\hat{z}$ to

generate the output o_{hc}. A relationship between the output o_{hc} and the quantized latent $\hat{z}$ can be described using Eq. 7.

$$o_{hc} = f_6(\hat{z};\ \theta_6) \qquad \text{Eq. 7}$$

where a parameter θ_6 represents parameters, such as weights used in convolution kernels in the hyper decoder (925) and biases (if biases are used in the hyper decoder (925)).

[0158] As described above, the compressed or encoded bits (932) can be added to the coded bitstream as the side information, which enables the entropy decoder (914) to use the conditional entropy model. Thus, the entropy model can be signal-dependent (e.g., block-dependent or image-dependent) and spatially adaptive, and thus can be more accurate than a fixed entropy model.

[0159] The NIC framework (900) can be suitably adapted, for example, to omit one or more components shown in FIG. 9B, to modify one or more components shown in FIG. 9B, and/or to include one or more components not shown in FIG. 9B. In an example, a NIC framework using a fixed entropy model includes the first sub-NN (951), and does not include the second sub-NN (952). In an example, a NIC framework includes the components in the NIC framework (900) except the entropy encoder (923) and the entropy decoder (924).

[0160] In an embodiment, one or more components in the NIC framework (900) shown in FIG. 9B are implemented using neural network(s), such as CNN(s). Each NN-based component (e.g., the main encoder network (911), the main decoder network (915), the context model NN (916), the entropy parameter NN (917), the hyper encoder (921), or the hyper decoder (925)) in a NIC framework (e.g., the NIC framework (900)) can include any suitable architecture (e.g., have any suitable combinations of layers), include any suitable types of parameters (e.g., weights, biases, a combination of weights and biases, and/or the like), and include any suitable number of parameters.

[0161] In an embodiment, the main encoder network (911), the main decoder network (915), the context model NN (916), the entropy parameter NN (917), the hyper encoder (921), and the hyper decoder (925) are implemented using respective CNNs.

[0162] FIG. 10 shows an exemplary CNN of the main encoder network (911) according to an embodiment of the disclosure. For example, the main encoder network (911) includes four sets of layers where each set of layers includes a convolution layer 5x5 c192 s2 followed by a GDN layer. One or more layers shown in FIG. 10 can be modified and /or omitted. Additional layer(s) can be added to the main encoder network (911).

[0163] FIG. 11 shows an exemplary CNN of the main decoder network (915) according to an embodiment of the disclosure. For example, the main decoder network (915) includes three sets of layers where each set of layers includes a deconvolution layer 5x5 c192 s2 followed by an IGDN layer. In addition, the three sets of layers are followed by a deconvolution layer 5x5 c3 s2 followed by an IGDN layer. One or more layers shown in FIG. 11 can be modified and /or omitted. Additional layer(s) can be added to the main decoder network (915).

[0164] FIG. 12 shows an exemplary CNN of the hyper encoder (921) according to an embodiment of the disclosure. For example, the hyper encoder (921) includes a convolution layer 3x3 c192 s1 followed by a leaky ReLU, a convolution layer 5x5 c192 s2 followed by a leaky ReLU, and a convolution layer 5x5 c192 s2. One or more layers shown in FIG. 12 can be modified and /or omitted. Additional layer(s) can be added to the hyper encoder (921).

[0165] FIG. 13 shows an exemplary CNN of the hyper decoder (925) according to an embodiment of the disclosure. For example, the hyper decoder (925) includes a deconvolution layer 5x5 c192 s2 followed by a leaky ReLU, a deconvolution layer 5x5 c288 s2 followed by a leaky ReLU, and a deconvolution layer 3x3 c384 s1. One or more layers shown in FIG. 13 can be modified and /or omitted. Additional layer(s) can be added to the hyper decoder (925).

[0166] FIG. 14 shows an exemplary CNN of the context model NN (916) according to an embodiment of the disclosure. For example, the context model NN (916) includes a masked convolution 5x5 c384 s1 for context prediction, and thus the context $\hat{y}_{<i}$ in Eq. 4 includes a limited context (e.g., a 5x5 convolution kernel). The convolution layer in FIG. 14 can be modified. Additional layer(s) can be added to the context model NN (916).

[0167] FIG. 15 shows an exemplary CNN of the entropy parameter NN (917) according to an embodiment of the disclosure. For example, the entropy parameter NN (917) includes a convolution layer 1x1 c640 s1 followed by a leaky ReLU, a convolution layer 1x1 c512 s1 followed by leaky ReLU, and a convolution layer 1x1 c384 s1. One or more layers shown in FIG. 15 can be modified and /or omitted. Additional layer(s) can be added to the entropy parameter NN (917).

[0168] The NIC framework (900) can be implemented using CNNs, as described with reference to FIGs. 10-15. The NIC framework (900) can be suitably adapted such that one or more components (e.g., (911), (915), (916), (917), (921), and/or (925)) in the NIC framework (900) are implemented using any suitable types of neural networks (e.g., CNNs or non-CNN

based neural networks). One or more other components the NIC framework (900) can be implemented using neural network(s).

[0169] The NIC framework (900) that includes neural networks (e.g., CNNs) can be trained to learn the parameters used in the neural networks. For example, when CNNs are used, the parameters represented by θ_1-θ_6, such as the weights used in the convolution kernels in the main encoder network (911) and biases (if biases are used in the main encoder network (911)), the weights used in the convolution kernels in the main decoder network (915) and biases (if biases are used in the main decoder network (915)), the weights used in the convolution kernels in the hyper encoder (921) and biases (if biases are used in the hyper encoder (921)), the weights used in the convolution kernels in the hyper decoder (925) and biases (if biases are used in the hyper decoder (925)), the weights used in the convolution kernel(s) in the context model NN (916) and biases (if biases are used in the context model NN (916)), and the weights used in the convolution kernels in the entropy parameter NN (917) and biases (if biases are used in the entropy parameter NN (917)), respectively, can be learned in the training process.

[0170] In an example, referring to FIG. 10, the main encoder network (911) includes four convolution layers where each convolution layer has a convolution kernel of 5x5 and 192 channels. Thus, a number of the weights used in the convolution kernels in the main encoder network (911) is 19200 (i.e., 4x5x5x192). The parameters used in the main encoder network (911) include the 19200 weights and optional biases. Additional parameter(s) can be included when biases and/or additional NN(s) are used in the main encoder network (911).

[0171] Referring to FIG. 9B, the NIC framework (900) includes at least one component or module built on neural network(s). The at least one component can include one or more of the main encoder network (911), the main decoder network (915), the hyper encoder (921), the hyper decoder (925), the context model NN (916), and the entropy parameter NN (917). The at least one component can be trained individually. In an example, the training process is used to learn the parameters for each component separately. The at least one component can be trained jointly as a group. In an example, the training process is used to learn the parameters for a subset of the at least one component jointly. In an example, the training process is used to learn the parameters for all of the at least one component, and thus is referred to as an E2E optimization.

[0172] In the training process for one or more components in the NIC framework (900), the weights (or the weight coefficients) of the one or more components can be

initialized. In an example, the weights are initialized based on pre-trained corresponding neural network model(s) (e.g., DNN models, CNN models). In an example, the weights are initialized by setting the weights to random numbers.

[0173] A set of training blocks can be employed to train the one or more components, for example, after the weights are initialized. The set of training blocks can include any suitable blocks having any suitable size(s). In some examples, the set of training blocks includes blocks from raw images, natural images, computer-generated images, and/or the like that are in the spatial domain. In some examples, the set of training blocks includes blocks from residue blocks or residue images having residue data in the spatial domain. The residue data can be calculated by a residue calculator (e.g., the residue calculator (723)). In some examples, raw images and/or residue images including residue data can be used directly to train neural networks in a NIC framework, such as the NIC framework (900). Thus, raw images, residue images, blocks from raw images, and/or blocks from residue images can be used to train neural networks in a NIC framework.

[0174] For purposes of brevity, the training process below is described using a training block as an example. The description can be suitably adapted to a training image. A training block t of the set of training blocks can be passed through the encoding process in FIG. 9B to generate a compressed representation (e.g., encoded information, for example, to a bitstream). The encoded information can be passed through the decoding process described in FIG. 9B to compute and reconstruct a reconstructed block $\bar{t}$.

[0175] For the NIC framework (900), two competing targets, e.g., a reconstruction quality and a bit consumption are balanced. A quality loss function (e.g., a distortion or distortion loss) $D(t, \bar{t})$ can be used to indicate the reconstruction quality, such as a difference between the reconstruction (e.g., the reconstructed block $\bar{t}$) and an original block (e.g., the training block t). A rate (or a rate loss) R can be used to indicate the bit consumption of the compressed representation. In an example, the rate loss R further includes the side information, for example, used in determining a context model.

[0176] For neural image compression, differentiable approximations of quantization can be used in E2E optimization. In various examples, in the training process of neural network-based image compression, noise injection is used to simulate quantization, and thus quantization is simulated by the noise injection instead of being performed by a quantizer (e.g., the quantizer (912)). Thus, training with noise injection can approximate the quantization error variationally. A bits per pixel (BPP) estimator can be used to simulate an

entropy coder, and thus entropy coding is simulated by the BPP estimator instead of being performed by an entropy encoder (e.g., (913)) and an entropy decoder (e.g., (914)). Therefore, the rate loss R in the loss function L shown in Eq. 1 during the training process can be estimated, for example, based on the noise injection and the BPP estimator. In general, a higher rate R can allow for a lower distortion D, and a lower rate R can lead to a higher distortion D. Thus, a trade-off hyperparameter λ in Eq. 1 can be used to optimize a joint R-D loss L where L as a summation of λD and R can be optimized. The training process can be used to adjust the parameters of the one or more components (e.g., (911) (915)) in the NIC framework (900) such that the joint R-D loss L is minimized or optimized. In some examples, a trade-off hyperparameter λ can be used to optimize the joint Rate-Distortion (R-D) loss as:

$$\mathrm{L}(x, \bar{x}, \hat{r}_1, \ldots, \hat{r}_N, \hat{y}) = \lambda D(x, \bar{x}) + \mathrm{R}(\textstyle\sum_1^n s_i, \sum_1^n u_i) + \beta E \qquad \text{Eq. 8}$$

where E measures the distortion of the decoded block residuals compared with the original block residuals before encoding, which acts as regularization loss for the residual encoding/decoding DNNs and the encoding/decoding DNNs. β is a hyperparameter to balance the importance of the regularization loss.

[0177] Various models can be used to determine the distortion loss D and the rate loss R, and thus to determine the joint R-D loss L in Eq. 1. In an example, the distortion loss $D(t, \bar{t})$ is expressed as a peak signal-to-noise ratio (PSNR) that is a metric based on mean squared error, a multiscale structural similarity (MS-SSIM) quality index, a weighted combination of the PSNR and MS-SSIM, or the like.

[0178] In an example, the target of the training process is to train the encoding neural network (e.g., the encoding DNN), such as a video encoder to be used on an encoder side and the decoding neural network (e.g., the decoding DNN), such as a video decoder to be used on a decoder side. In an example, referring to FIG. 9B, the encoding neural network can include the main encoder network (911), the hyper encoder (921), the hyper decoder (925), the context model NN (916), and the entropy parameter NN (917). The decoding neural network can include the main decoder network (915), the hyper decoder (925), the context model NN (916), and the entropy parameter NN (917). The video encoder and/or the video decoder can include other component(s) that are based on NN(s) and/or not based on NN(s).

[0179] The NIC framework (e.g., the NIC framework (900)) can be trained in an E2E fashion. In an example, the encoding neural network and the decoding neural network are updated jointly in the training process based on backpropagated gradients in an E2E fashion.

[0180] After the parameters of the neural networks in the NIC framework (900) are trained, one or more components in the NIC framework (900) can be used to encode and/or decode blocks. In an embodiment, on the encoder side, the video encoder is configured to encode the input signal x into the encoded signal (931) to be transmitted in the bitstream. The video encoder can include multiple components in the NIC framework (900). In an embodiment, on the decoder side, the corresponding video decoder is configured to decode the encoded signal (931) in the bitstream into the reconstructed signal $\bar{x}$. The video decoder can include multiple components in the NIC framework (900).

[0181] In an example, the video encoder includes all the components in the NIC framework (900), for example, when content-adaptive online training is employed.

[0182] FIG. 16A shows an exemplary video encoder (1600A) according to an embodiment of the disclosure. The video encoder (1600A) includes the main encoder network (911), the quantizer (912), the entropy encoder (913), and the second sub-NN (952) that are described with reference to FIG. 9B for example. FIG. 16B shows an exemplary video decoder (1600B) according to an embodiment of the disclosure. The video decoder (1600B) can correspond to the video encoder (1600A). The video decoder (1600B) can include the main decoder network (915), the entropy decoder (914), the context model NN (916), the entropy parameter NN (917), the entropy decoder (924), and the hyper decoder (925). Referring to FIGs. 16A-16B, on the encoder side, the video encoder (1600A) can generate the encoded signal (931) and the encoded bits (932) to be transmitted in the bitstream. On the decoder side, the video decoder (1600B) can receive and decode the encoded signal (931) and the encoded bits (932).

[0183] FIGs. 17-18 show an exemplary video encoder (1700) and a corresponding video decoder (1800), respectively, according to embodiments of the disclosure. Referring to FIG. 17, the encoder (1700) includes the main encoder network (911), the quantizer (912), and the entropy encoder (913). Examples of the main encoder network (911), the quantizer (912), and the entropy encoder (913) are described with reference to FIG. 9B. Referring to FIG. 18, the video decoder (1800) includes the main decoder network (915) and the entropy decoder (914). Examples of the main decoder network (915) and the entropy decoder (914) are described with reference to FIG. 9B. Referring to FIGs. 17 and 18, the video encoder (1700) can generate the encoded signal (931) to be included in the bitstream. The video decoder (1800) can receive and decode the encoded signal (931).

[0184] Post-filtering or post-processing can be applied to a reconstructed signal, such as a reconstructed block or a reconstructed image, to determine a post-filtered signal (e.g., a post-filtered image or a post-filtered block). Post-filtering can be applied to reduce a distortion loss of the post-filtered signal, for example, the distortion loss of the post-filtered signal can be less than a distortion loss of the corresponding reconstructed signal.

[0185] FIG. 19A shows an example of block-wise image filtering according to an embodiment of the disclosure. Instead of directly post-filtering an entire image (1980), a block-based filter mechanism can be applied for post-filtering. The entire image (1980) is partitioned into blocks (1981)-(1996) of a same size (e.g., WxH samples) or various sizes, and the blocks (1981)-(1996) are post-filtered individually. In an embodiment, a post-filtering module is applied to the blocks (1981)-(1996) instead of the image (1980).

[0186] In an embodiment, an image (e.g., a reconstructed image) is post-filtered without being split into blocks, and thus the entire image is an input to a post-filtering module.

[0187] FIG. 19B shows an exemplary post-filtering module (1910) and a NIC framework (e.g., the NIC framework (900)) according to an embodiment of the disclosure. The NIC framework (900) is described with reference to FIG. 9B. The reconstructed signal $\bar{x}$ from the main decoder network (915) can be fed into the post-filtering module (1910), and a post-filtered signal $\bar{\bar{x}}$ can be generated by the post-filtering module (1910). In an example, the distortion loss $D(x, \bar{x})$ of the reconstructed block $\bar{x}$ is reduced to a distortion loss $D(x, \bar{\bar{x}})$ of the post-filtered signal $\bar{\bar{x}}$ where the distortion loss $D(x, \bar{\bar{x}})$ of the post-filtered signal $\bar{\bar{x}}$ is less than the distortion loss $D(x, \bar{x})$ of the reconstructed block $\bar{x}$.

[0188] The post-filtering module (1910) can be implemented using any suitable filtering methods that can reduce the distortion loss and improve visual quality. According to an embodiment of the disclosure, the post-filtering module (1910) is NN-based and is implemented using a NN (e.g., a post-filtering NN), such as a post-filtering DNN or a post-filtering CNN.

[0189] The post-filtering NN can include any suitable architecture (e.g., have any suitable combinations of layers), include any suitable types of parameters (e.g., weights, biases, a combination of weights and biases, and/or the like), and include any suitable number of parameters, as described in the disclosure.

[0190] In an example, the post-filtering NN is implemented using a CNN including one or more convolutional layers. The post-filtering NN can include additional layer(s)

described in the disclosure, such as pooling layer(s), fully connected layer(s), normalization layer(s), and/or the like. The layers in the post-filtering NN can be arranged in any suitable order and in any suitable architecture (e.g., a feed-forward architecture or a recurrent architecture). In an example, a convolutional layer is followed by other layer(s), such as pooling layer(s), fully connected layer(s), normalization layer(s), and/or the like. Each of the convolutional layers in the post-filtering NN can include any suitable number of channels and any suitable convolution kernels and strides.

[0191] The post-filtering NN can be trained based on training signals (e.g., training images and/or training blocks), similar or identical to those described above with reference to FIG. 9B. In some examples, an input signal (e.g., an input image or an input block) to be compressed (e.g., encoded) and/or transmitted has properties that are significantly different from the training signals. Thus, post-filtering the reconstructed signal that corresponds to the input signal using the post-filtering NN trained with the training signals can lead to a relatively poor R-D loss L (e.g., a relatively large distortion). Therefore, aspects of the disclosure describe a content-adaptive online training method for post-filtering. In the content-adaptive online training method for post-filtering, one or more parameters of the post-filtering NN can be determined based on one or more input signals to be compressed (e.g., encoded) and/or transmitted. The input signal(s) can include raw data or residue data.

[0192] In an embodiment, the one or more input signals are one or more blocks in an input image. The one or more blocks can be used to determine the one or more parameters in the post-filtering NN by optimizing rate-distortion performance, and the content-adaptive online training method for post-filtering can be referred to as the block-wise content-adaptive online training method for post-filtering.

[0193] In an embodiment, the one or more input signals are one or more input images. The one or more input images can be used to determine the one or more parameters in the post-filtering NN by optimizing rate-distortion performance, and the content-adaptive online training method for post-filtering can be referred to as the image-based content-adaptive online training method for post-filtering.

[0194] Post-filtering information indicating the determined one or more parameters can be encoded into a video bitstream along with the encoded one or more signals (e.g., the encoded one or more blocks or the encoded one or more images). At a decoder side, the post-filtering NN can be determined based on the determined one or more parameters, and can post-filter the reconstructed one or more signals and can achieve better compression

performance by using the determined one or more parameters. The reconstructed one or more signals can be reconstructed by the video decoder based on the encoded one or more signals, respectively.

[0195] According to aspects of the disclosure, content-adaptive online training can be applied in post-filtering, such as NN-based post-filtering. The NN-based post-filtering (e.g., DNN-based post-filtering or CNN-based post-filtering) can be implemented in an NN-based image coding framework (e.g., the NIC framework (900)), or other image compression methods.

[0196] The post-filtering module (1910) can be implemented using a NN (e.g., a DNN or a CNN) and can be applied to the reconstructed signal $\bar{x}$. The NN in the post-filtering module can be referred to as a post-filtering NN, such as a post-filtering DNN or a post-filtering CNN. An output of the post-filtering module (1910) is the post-filtered signal $\bar{\bar{x}}$. According to an embodiment of the disclosure, the content-adaptive online training of the post-filtering module (1910) can be implemented to reduce a R-D loss $L_p(x, \bar{\bar{x}}, \hat{x})$ shown in Eq. 9.

$$L_p(x, \bar{\bar{x}}, \hat{x}) = \lambda D(x, \bar{\bar{x}}) + \mathrm{R}(\hat{x}) + \mathrm{R(p)} \qquad \text{Eq. 9}$$

where the R-D loss $L_p(x, \bar{\bar{x}}, \hat{x})$ corresponding to the post-filtered signal $\bar{\bar{x}}$ can include the distortion loss $D(x, \bar{\bar{x}})$ of the post-filtered signal $\bar{\bar{x}}$, the rate loss R $(\hat{x})$ of the encoded signal $\hat{x}$, and a rate loss or a bit consumption R(p) of signaling post-filtering information in a coded video bitstream. The post-filtering information can indicate the determined one or more parameters of the post-filtering NN.

[0197] For purposes of brevity, the content-adaptive online training of the post-filtering module (1910) can be referred to as the post-filtering training. The block-wise content-adaptive online training method for post-filtering can be referred to as the block-wise post-filtering training. The image-based content-adaptive online training method for post-filtering can be referred to as the image-based post-filtering training.

[0198] In an embodiment, a goal of the post-filtering training is to decrease (e.g., minimize) the R-D loss L_p such that the R-D loss L_p is less than the R-D loss $L(x, \bar{x}, \hat{x})$ of the reconstructed signal $\bar{x}$ as described in Eq. 1. Comparing Eqs. 1 and 8, the rate loss R $(\hat{x})$ remains unchanged. Since the rate loss R (p) increases the R-D loss L_p, the post-filtering training can significantly decrease the distortion loss $D(x, \bar{\bar{x}})$ such that $D(x, \bar{\bar{x}}) < D(x, \bar{x})$.

[0199] According to an embodiment of the disclosure, the one or more parameters in the post-filtering NN can be determined in the post-filtering training, for example, based on

one or more input signals. Referring to FIG. 19B, the input signal x is fed into the NIC framework (900), and the encoded signal $\hat{x}$ is generated in the encoding process. The reconstructed signal $\bar{x}$ is generated in the decoding process. In an embodiment, the one or more parameters in the post-filtering NN can be determined in the post-filtering training based on the reconstructed signal $\bar{x}$. In an example, the one or more parameters in the post-filtering NN are iteratively updated based on the reconstructed signal $\bar{x}$ to reduce the distortion loss $D(x, \bar{\bar{x}})$. Thus, the post-filtering NN is dependent on the reconstructed signal $\bar{x}$, which is dependent on the input signal x that is to be coded.

[0200] The post-filtering NN can be configured with initial parameters (e.g., initial weights and/or initial biases), for example, prior to the post-filtering training. In an embodiment, the post-filtering NN is pretrained based on a training dataset including training blocks and/or training images, and the initial parameters include pretrained parameters (e.g., pretrained weights and/or pretrained biases). In an embodiment, the post-filtering NN is not pretrained. The initial parameters can be random parameters.

[0201] In an embodiment, one or more initial parameters of the initial parameters in the post-filtering NN are iteratively updated in the post-filtering training based on the reconstructed signal $\bar{x}$ which is determined based on the input signal x. The one or more initial parameters can be updated (e.g., replaced) by the one or more parameters (e.g., one or more replacement parameters) determined in the post-filtering training. In an example, an entire set of the initial parameters is updated. In an example, a subset of the initial parameters is updated by the one or more replacement parameters and a remaining subset of the initial parameters remains unchanged by the post-filtering training.

[0202] The post-filtering training can be used as a preprocessing step (e.g., a pre-encoding step) for boosting the compression performance of any image compression method.

[0203] The determined one or more parameters can be encoded into the video bitstream, along with the encoded signal $\hat{x}$. The rate loss of the encoded signal $\hat{x}$ is represented by the rate loss R ($\hat{x}$), and the rate loss of the encoded one or more parameters is represented by the rate loss R (p). The post-filtering training can be agnostic to an image compression codec, and can be implemented with any suitable type of image compression codec. The image compression codec can be NN-based, such as the NIC framework (900) shown in FIGs. 9B and 19B. The image compression codec can be implemented without a NN, such as in certain implementations of FIGs. 5-8.

[0204] By using the determined one or more parameters (e.g., the one or more replacement parameters) in the post-filtering NN, the post-filtering module (1910) can be applied to the reconstructed signal $\bar{x}$ (e.g., the reconstructed image or the reconstructed block) and achieve a better distortion performance, such as a smaller R-D loss L_p. As described with reference to Eq. 8, R(p) represents the bit consumption of the determined one or more parameters (e.g., the one or more replacement parameters) encoded into the video bitstream.

[0205] The above descriptions can be suitably adapted when a plurality of input signals is used to determine the one or more parameters where the determined one or more parameters are shared by the plurality of input signals.

[0206] Referring to FIG. 19B, at an encoder side, the one or more parameters of the post-filtering NN in the post-filtering module (1910) can be determined based on the reconstructed signal $\bar{x}$ in the post-filtering training. The encoded signal $\hat{x}$ can be generated based on the input signal x. In an embodiment, the post-filtering information indicating the determined one or more parameters of the post-filtering NN can be encoded. The post-filtering information corresponds to the encoded signal $\hat{x}$. The encoded signal $\hat{x}$ and the post-filtering information can be included in the coded video bitstream.

[0207] FIG. 20A shows an exemplary video decoder (2000A) according to an embodiment of the disclosure. The video decoder (2000A) can include components of the video decoder (1600B) that are described with reference to FIG. 16B and the post-filtering module (1910) that is described with reference to FIG. 19B.

[0208] FIG. 20B shows an exemplary video decoder (2000B) according to an embodiment of the disclosure. The video decoder (2000B) can include components of the video decoder (1800) that are described with reference to FIG. 18 and the post-filtering module (1910) that is described with reference to FIG. 19B.

[0209] At a decoder side, referring to FIGs. 19B, 20A, and 20B, the post-filtering information can be decoded by the video decoder (e.g., the video decoder (2000A) or (2000B)). The one or more parameters of the post-filtering NN in the post-filtering module (1910) can be obtained based on the post-filtering information. In an embodiment, the post-filtering NN is configured with initial parameters, and the one or more parameters are the one or more replacement parameters. One or more of the initial parameters can be updated with the one or more replacement parameters, respectively.

[0210] In an example, the post-filtering information indicates differences between the one or more parameters and the respective one or more of the initial parameters. The one or

more parameters can be determined according to a sum of the differences and the respective one or more of the initial parameters.

[0211] The post-filtering NN in the post-filtering module (1910) that is in the video decoder (e.g., the video decoder (2000A) or (2000B)) can be determined based on the post-filtering information. In an embodiment, the one or more initial parameters in the post-filtering NN are updated based on the one or more parameters, for example, the one or more initial parameters are replaced by the one or more parameters.

[0212] Referring to FIGs. 19B, 20A, and 20B, the encoded signal $\hat{x}$ can be decoded by the video decoder (2000A) or (2000B). For example, the encoded signal $\hat{x}$ is reconstructed by the main decoder network (915) to generate the reconstructed signal $\bar{x}$. The reconstructed signal $\bar{x}$ can be post-filtered based on the post-filtering NN that is determined (e.g., updated) based on the one or more parameters. In an example, the reconstructed signal $\bar{x}$ is post-filtered based on the updated post-filtering NN.

[0213] In some examples, the one or more parameters indicated by the post-filtering information are decompressed and then used to update the post-filtering NN.

[0214] According to an embodiment of the disclosure, the post-filtering training can be referred to as a finetuning process where the one or more initial parameters (e.g., the one or more pretrained parameters) of the initial parameters in the post-filtering NN can be updated (e.g., finetuned) based on the input signal(s) to be encoded and/or transmitted. The input signal(s) can be different from the training signals used to obtain the initial parameters. Thus, the post-filtering NN can be adapted to target the content of the input signal(s).

[0215] In an example, the input signal(s) includes one or more input signals to be encoded and/or transmitted. In an example, the one or more input signals include a single input signal (e.g., a single input image or a single input block), and the post-filtering training is performed with the single input signal. The post-filtering NN is determined (e.g., trained or updated) based on the single input signal. The determined post-filtering NN on a decoder side can be used to post-filter the reconstructed signal (e.g., the reconstructed image or the reconstructed block) corresponding to the single input signal and optionally other reconstructed signals corresponding to other input signals. The post-filtering information can be encoded into the video bitstream together with the encoded single input signal (e.g., the encoded image or the encoded block).

[0216] In an embodiment, the one or more input signals include multiple input signals, and the post-filtering training is performed with the multiple input signals. The post-

filtering NN is determined (e.g., trained or updated) based on the multiple input signals. The determined post-filtering NN on the decoder side can be used to post-filter the reconstructed signals corresponding to the multiple input signals and optionally other reconstructed signals corresponding to other input signals. The post-filtering information can be encoded into the video bitstream together with the encoded multiple input signals.

[0217] The rate loss R_p can increase with the signaling of the post-filtering information in the video bitstream. When the input signal(s) includes the single input signal, the post-filtering information is signaled for each encoded signal, and a first increase to the rate loss R_p is used to indicate the increase to the rate loss R_p due to the signaling of the post-filtering information per signal. When the input signal(s) includes the multiple input signals, the post-filtering information is signaled for and shared by the multiple input signals, and a second increase to the rate loss R_p is used to indicate the increase to the rate loss R_p due to the signaling of the post-filtering information per signal. Because the post-filtering information is shared by the multiple input signals, the second increase to the rate loss R_p can be less than the first increase to the rate loss R_p. Thus, in some examples, it can be advantageous to determine (e.g., train or update) the post-filtering NN using the multiple input signals.

[0218] The post-filtering NN can be specified by different types of parameters, such as weights, biases, and the like. The post-filtering NN can be configured with suitable initial parameters, such as weights, biases, or a combination of weights and biases. When CNN(s) are used, the weights can include elements in convolution kernels. As described above, the subset or the entire set of the initial parameters can be updated by the post-filtering training.

[0219] In an embodiment, the one or more initial parameters that are updated are in a single layer (e.g., a single convolutional layer) of the post-filtering NN. In an embodiment, the one or more initial parameters that are updated are in multiple or all layers (e.g., multiple or all convolutional layers) of the post-filtering NN.

[0220] One or more types of parameters can be used to specify the post-filtering NN. In an embodiment, the one or more initial parameters to be updated are bias term(s), and only the bias term(s) are replaced by the determined one or more parameters. In an embodiment, the one or more initial parameters to be updated are weights, and only the weights are replaced by the determined one or more parameters. In an embodiment, the one or more initial parameters to be updated include the weights and bias term(s) and are replaced by the determined one or more parameters. In an example, the entire set of the initial parameters is replaced by the determined one or more parameters.

[0221] The post-filtering training can include multiple epochs (e.g., iterations) where the one or more initial parameters are updated in an iterative process. The post-filtering training can stop when a training loss has flattened or is about to flatten. In an example, the post-filtering training stops when the training loss (e.g., the R-D loss L_p) is below a first threshold. In an example, the post-filtering training stops when a difference between two successive training losses is below a second threshold.

[0222] Two hyperparameters (e.g., a step size and a maximum number of steps) can be used in the post-filtering training together with a loss function (e.g., the R-D loss L_p). The maximum number of iterations can be used as a threshold of a maximum number of iterations to terminate the post-filtering training. In an example, the post-filtering training stops when a number of iterations reaches the maximum number of iterations.

[0223] The step size can indicate a learning rate of the online training process (e.g., the post-filtering training). The step size can be used in a gradient descent algorithm or a backpropagation calculation performed in the post-filtering training. A step size can be determined using any suitable method.

[0224] In the block-wise post-filtering training, the step size for each block in an image can be different. In an embodiment, different step sizes can be assigned for an image in order to achieve a better compression result (e.g., a better R-D loss Lp).

[0225] In the image-based post-filtering training, the step size for each image can be different. In an embodiment, different step sizes can be assigned for different input images in order to achieve a better compression result (e.g., a better R-D loss Lp).

[0226] In an embodiment, different step sizes are used for signals (e.g., blocks or images) with different types of contents to achieve optimal results. The signals can include the input signals (e.g., the input signal x), the encoded signals (e.g., the encoded signal $\hat{x}$), the reconstructed signals (e.g., the reconstructed signal $\bar{x}$), or the like. The relationships of the input signal (e.g., the input signal x), the encoded signal (e.g., the encoded signal $\hat{x}$), and the corresponding reconstructed signal (e.g., the reconstructed signal $\bar{x}$) are described with reference to FIGs. 9B and 19B. Different types can refer to different variances. In an example, the step size is determined based on a variance of a signal (e.g., a block or an image) used to update the post-filtering NN. For example, a step size of a signal having a high variance is larger than a step size of a signal having a low variance where the high variance is larger than the low variance.

[0227] In an embodiment, a step size is chosen based on characteristics of a signal (e.g., a block or an image), such as RGB variance of the signal. In an embodiment, a step size is chosen based on RD performance (e.g., a R-D loss L) of the signal (e.g., the block or the image). Multiple sets of parameter(s) (e.g., multiple sets of replacement parameter(s)) can be generated based on different step sizes, and the set with the better compression performance (e.g., a smaller R-D loss) can be chosen.

[0228] In an embodiment, a first step size can be used to run a certain number (e.g., 100) of iterations. Then, a second step size (e.g., the first step size plus or minus a size increment) can be used to run the certain number of iterations. Results from the first step size and the second step size can be compared to determine a step size to be used. More than two step sizes may be tested to determine an optimal step size.

[0229] A step size can vary during the post-filtering training. The step size can have an initial value at an onset of the post-filtering training, and the initial value can be reduced (e.g., halved) at a later stage of the post-filtering training, for example, after a certain number of iterations to achieve a finer tuning. The step size or the learning rate can be varied by a scheduler during the iterative post-filtering training. The scheduler can include a parameter adjustment method used to adjust the step size. The scheduler can determine a value for the step size such that the step size can increase, decrease, or remain constant in a number of intervals. In an example, the learning rate is altered in each step by the scheduler. A single scheduler or multiple different schedulers can be used for different blocks or different images. Thus, multiple sets of parameter(s) can be generated based on the multiple schedulers, and one of the multiple sets of parameter(s) with the better compression performance (e.g., a smaller R-D loss L_p) can be chosen.

[0230] In an embodiment, multiple learning rate schedules are assigned for different signals (e.g., different blocks or different images) in order to achieve better compression result. In an example, all blocks in an image share a same learning rate schedule. In an example, a set of images share a same learning rate schedule. In an embodiment, selection of learning rate schedules is based on characteristics of a signal (e.g., a block or an image), such as a RGB variance of the signal. In an embodiment, selection of learning rate schedules is based on the RD performance of the signal.

[0231] In an embodiment, different signals (e.g., different blocks or different images) can be used to determine (e.g., update) different types of parameters (e.g., biases or weights) in the post-filtering NN. For example, a first signal (e.g., a first block) is used to update at

least one bias in the post-filtering NN, and a second signal (e.g., a second block) is used to update at least one weight in the post-filtering NN.

[0232] In an embodiment, multiple signals (e.g., multiple blocks in an image or multiple images) are used to determine (e.g., update) the same one or more parameters. In an example, all the blocks in an image are used to determine (e.g., update) the same one or more parameters.

[0233] In an embodiment, the one or more initial parameters to be updated are chosen based on a characteristic of a signal (e.g., a block or an image), such as a RGB variance of the signal. In an embodiment, the one or more initial parameters to be updated are chosen based on a RD performance of the signal.

[0234] At the end of the post-filtering training, one or more updated parameters can be computed for the respective determined one or more parameters (e.g., the respective one or more replacement parameters). In an embodiment, the one or more updated parameters are calculated as differences between the determined one or more parameters (e.g., the respective one or more replacement parameters) and the corresponding one or more initial parameters (e.g., the one or more pretrained parameters). In an embodiment, the one or more updated parameters are the determined one or more parameters, respectively.

[0235] In an embodiment, how to obtain the one or more updated parameters from the respective determined one or more parameters depends on the signal (e.g., the input signal x, the reconstructed signal $\bar{x}$) used in the post-filtering training. Different methods can be used for different signals. In an example, an updated parameter for a post-filtering NN applied to a first block is calculated as a difference between a replacement parameter obtained based on the first block and a corresponding pretrained parameter. In an example, an updated parameter for a post-filtering NN applied to a second block is a replacement parameter obtained based on the second block.

[0236] In an embodiment, different blocks have different relationships between the one or more updated parameters and the one or more replacement parameters. For example, for a first block the one or more updated parameters are calculated as differences between the one or more replacement parameters and the corresponding one or more pretrained parameters. For a second block, the one or more updated parameters are the one or more replacement parameters, respectively.

[0237] In an embodiment, how to obtain the one or more updated parameters from the respective determined one or more parameters does not depend on the signal (e.g., the input

signal x, the reconstructed signal $\bar{x}$) used in the post-filtering training. In an example, all blocks share the same way to update parameter(s) in the post-filtering NN. In an embodiment, multiple blocks (e.g., all blocks) in an image have a same relationship between the one or more updated parameters and the one or more replacement parameters.

[0238] In an embodiment, the relationship between the one or more updated parameters and the one or more replacement parameters is chosen based on characteristics of a signal (e.g., a block or an image), such as a RGB variance of the signal. In an embodiment, the relationship between the one or more updated parameters and the one or more replacement parameters is chosen based on a RD performance of the signal.

[0239] In an embodiment, the one or more updated parameters can be generated from the determined one or more parameters (e.g., the one or more replacement parameters), for example, using a certain linear or nonlinear transform, and the one or more updated parameters are representative parameter(s) generated based on the determined one or more parameters. The determined one or more parameters are transformed into the one or more updated parameters for better compression.

[0240] In an example, the one or more updated parameters can be compressed, for example, using LZMA2 that is a variation of a Lempel–Ziv–Markov chain algorithm (LZMA), a bzip2 algorithm, or the like. In an example, compression is omitted for the one or more updated parameters. In some embodiments, the one or more updated parameters can be encoded into the video bitstream as the post-filtering information where the post-filtering information can indicate the determined one or more parameters (e.g., the one or more replacement parameters).

[0241] In an embodiment, compression methods for the one or more updated parameters are different for different signals (e.g., different blocks or different images). For example, for a first block, the LZMA2 is used to compress the one or more updated parameters, and for a second block, the bzip2 is used to compress the one or more updated parameters. In an embodiment, a same compression method is used to compress the one or more updated parameters for multiple blocks (e.g., all blocks) in an image. In an embodiment, a compression method is chosen based on characteristics of a signal (e.g., a block or an image), such as a RGB variance of the signal. In an embodiment, a compression method is chosen based on RD performance of the signal.

[0242] In an embodiment, a structure (e.g., an architecture) of the post-filtering NN for each signal is identical. The structure of the post-filtering NN can include a number of

layers, how different nodes and/or layers are organized and connected, a feed-forward architecture, a recurrent architecture, a DNN, a CNN, and/or the like. In an example, the structure refers to a number of convolution layers, and different blocks have the same number of convolution layers.

[0243] In an embodiment, different structures of the post-filtering NN correspond to different signals. In an example, the post-filtering NN used to post-filter different blocks has different numbers of convolution layers.

[0244] In an embodiment, whether to use the post-filtering module to post-filter a signal (e.g., a block or an image) is determined based on a comparison of the R-D loss L_p and the R-D loss L. In an embodiment, a post-filtering module with the best R-D performance (e.g., the smallest R-D loss L_p) is chosen based on a comparison of different R-D loss L_{ps} of different post-filtering modules.

[0245] According to an embodiment of the disclosure, each signal (e.g., each block or each image) corresponds to a post-filtering NN that is determined in the post-filtering training based on the signal. A post-filtering NN can be updated independently from another post-filtering NN. For example, the post-filtering NN corresponding to a first signal is updated independently from the post-filtering NN corresponding to a second signal.

[0246] According to an embodiment of the disclosure, the post-filtering NN corresponding to a signal (e.g., the second signal) can be updated based on the post-filtering NN corresponding to another signal (e.g., the first signal). In an example, first post-filtering information in a coded bitstream corresponding to the first signal is decoded. The first post-filtering information can indicate a first parameter of the post-filtering NN in the video decoder. The post-filtering NN in the video decoder corresponding to the first signal can be determined based on the first parameter indicated by the first post-filtering information. In an example, the first signal includes a first encoded block that is to be reconstructed. The first signal can be decoded based on the determined post-filtering NN corresponding to the first signal. For example, the first signal is reconstructed and the reconstructed first signal is post-filtered based on the determined post-filtering NN.

[0247] Parameter(s) in the post-filtering NN corresponding to the first signal can be used to update the post-filtering NN corresponding to the second signal. For example, pixel distributions in blocks of a same image can be similar, and thus parameter(s) to be updated of the post-filter NN corresponding to different blocks may be reduced. According to an embodiment of the disclosure, second post-filtering information in the coded bitstream

corresponding to the second signal can be decoded. The second post-filtering information can indicate a second parameter. The second signal is different from the first signal. The post-filtering NN corresponding to the first signal can be updated based on the second parameter. The updated post-filtering NN corresponds to the second signal and is configured with the first parameter and the second parameter. The second signal can be decoded (e.g., post-filtered) based on the updated post-filtering NN corresponding to the second signal.

[0248] In an embodiment, different post-filtering NNs can be applied to signals (e.g., blocks or images) having different sizes. In general, a number of parameters in a post-filtering NN can increase with a size (e.g., a width, a height, or an area) of a signal (e.g., a block or an image).

[0249] In an embodiment, different post-filtering NNs are applied to a same signal (e.g., an image or a block) corresponding to different compression quality targets.

[0250] The NIC framework and the post-filtering NN can include any type of neural networks and use any neural network-based image compression methods, such as a context-hyperprior encoder-decoder framework, a scale-hyperprior encoder-decoder framework, a Gaussian Mixture Likelihoods framework and variants of the Gaussian Mixture Likelihoods framework, an RNN-based recursive compression method and variants of the RNN-based recursive compression method, and the like.

[0251] Compared with related E2E image compression methods, the post-filtering training methods and apparatus in the disclosure can have the following benefits. Adaptive online training mechanisms are exploited to improve the coding efficiency. Use of a flexible and general framework can accommodate various types of pretrained frameworks and quality metrics. For example, certain pretrained parameters in the post-filtering NN can be replaced by using online training with block(s) or image(s) to be encoded and transmitted.

[0252] Video coding technologies can include filtering operations that are performed on reconstructed samples so that artifacts resulting from lossy compression, such as by quantization, can be reduced. A deblocking filter process is used in one such filtering operation, in which a block boundary (e.g., a boundary region) between two adjacent blocks can be filtered so that a smoother transition of sample values from one block to the other block can be achieved.

[0253] In some related examples (e.g., HEVC), the deblocking filter process can be applied to samples adjacent to the block boundary. The deblocking filter process can be performed for each CU in the same order as the decoding process. For example, the

deblocking filter process can be performed by horizontal filtering for vertical boundaries for an image first, followed by vertical filtering for horizontal boundaries for the image. Filtering can be applied to 8×8 block boundaries which are determined to be filtered, both for luma and chroma components. In an example, 4×4 block boundaries are not processed in order to reduce the complexity.

[0254] A boundary strength (BS) can be used to indicate a degree or strength of the deblocking filter process. In an embodiment, a value of 2 for BS indicates strong filtering, 1 indicates weak filtering, and 0 indicates no deblocking filtering.

[0255] Blocks of an image can be reconstructed from a coded video bitstream using any suitable methods, such as the embodiments in the disclosure. For example, the blocks can be reconstructed using a video decoder (e.g., (1600B), (1800), (2000A), or (2000B)), for example, including neural networks (e.g., CNN(s)). The blocks can be reconstructed using a video decoder that is not based on NNs. According to some embodiments of the disclosure, post-filtering or post-processing can be performed on one of a plurality of regions of first two neighboring reconstructed blocks of the reconstructed blocks with at least one post-processing NN. The first two neighboring reconstructed blocks can have a first shared boundary and include a boundary region having samples on both sides of the first shared boundary. The plurality of regions of the first two neighboring reconstructed blocks can include the boundary region and non-boundary regions that are outside the boundary region. The one of the plurality of regions can be replaced with the post-processed one of the plurality of regions of the first two neighboring reconstructed blocks. The post-processing that is performed can be deblocking the boundary region, enhancing one or more of the non-boundary regions, a combination of deblocking and enhancing, and/or the like.

[0256] One or more deblocking methods can be used to reduce artifacts among blocks (e.g., the reconstructed blocks in the image). To reduce the artifacts among the blocks, such as artifacts in boundary regions, one or more NN-based deblocking models can be used. The NN-based deblocking models can be DNN-based deblocking models, CNN-based deblocking models, or the like. The NN-based deblocking models can be implemented using NNs, such as DNNs, CNNs, or the like.

[0257] In an embodiment, the one of the plurality regions is the boundary region. The at least one post-processing NN includes at least one deblocking NN, and deblocking can be performed on the boundary region with the at least one deblocking NN. The boundary region

can be replaced with the deblocked boundary region. Examples of deblocking are shown in FIGs. 21A-21C, 22, 23, and 26.

[0258] FIGs. 21A-21C show an exemplary deblocking process (2100) according to an embodiment of the disclosure. Referring to FIG. 21A, an image (2101) can be partitioned into a plurality of blocks (2111)-(2114). For brevity, four equal-sized blocks (2111)-(2114) are illustrated in FIG. 21A. In general, an image can be partitioned into any suitable number of blocks, and sizes of the blocks can be different or identical, and the description can be suitably adapted. In some examples, regions that include artifacts, for example, due to partitioning an image to blocks, can be processed by deblocking.

[0259] In an example, the blocks (2111)-(2114) are reconstructed blocks from the main decoder network (915). The first two neighboring reconstructed blocks of the reconstructed blocks (2111)-(2114) can include the blocks (2111) and (2113) separated by a first shared boundary (2141). The blocks (2111) and (2113) can include a boundary region A having samples on both sides of the first shared boundary (2141). Referring to FIGs. 21A-21B, the boundary region A can include sub-boundary regions A1 and A2 that are located in the blocks (2111) and (2113), respectively.

[0260] Two neighboring reconstructed blocks of the reconstructed blocks (2111)-(2114) can include the blocks (2112) and (2114) separated by a second shared boundary (2142). The blocks (2112) and (2114) can include a boundary region B having samples on both sides of the second shared boundary (2142). Referring to FIGs. 21A-21B, the boundary region B can include sub-boundary regions B1 and B2 that are located in the blocks (2112) and (2114), respectively.

[0261] Two neighboring reconstructed blocks of the reconstructed blocks (2111)-(2114) can include the blocks (2111) and (2112) separated by a shared boundary (2143). The blocks (2111) and (2112) can include a boundary region C having samples on both sides of the shared boundary (2143). Referring to FIGs. 21A-21B, the boundary region C can include sub-boundary regions C1 and C2 that are located in the blocks (2111) and (2112), respectively.

[0262] Two neighboring reconstructed blocks of the reconstructed blocks (2111)-(2114) can include the blocks (2113) and (2114) separated by a shared boundary (2144). The blocks (2113) and (2114) can include a boundary region D having samples on both sides of the shared boundary (2144). Referring to FIGs. 21A-21B, the boundary region D can include

sub-boundary regions D1 and D2 that are located in the blocks (2113) and (2114), respectively.

[0263] The sub-boundary regions A1-D1 and A2-D2 (and the boundary regions A-D) can have any suitable sizes (e.g., widths and/or heights). In an embodiment shown in FIG. 21A, the sub-boundary regions A1, A2, B1, and B2 have an identical size of m × n, where n is a width of the blocks (2111)-(2114), m is a height of the sub-boundary regions A1, A2, B1, and B2. Both m and n are positive integers. In an example, m is four pixels or four samples. Thus, the boundary regions A and B have an identical size of 2m × n. The sub-boundary regions C1, C2, D1, and D2 have an identical size of n × m, where n is a height of the blocks (2111)-(2114), m is a width of the sub-boundary regions C1, C2, D1, and D2. Thus, the boundary regions C and D have an identical size of n × 2m. As described above, the sub-boundary regions and the boundary regions can have different sizes, such as different widths, different heights, and/or the like. For example, the sub-boundary regions A1 and A2 can have different heights. In an example, the sub-boundary regions C1 and C2 can have different widths. The boundary regions A and B can have different widths. The boundary regions C and D can have different heights.

[0264] Referring to FIGs. 21A-21B, the boundary region A includes m lines of samples (e.g., m rows of samples) in the block (2111) from the first shared boundary (2141) and m lines of samples (e.g., m rows of samples) in the block (2113) from the first shared boundary (2141). The boundary region C includes m lines of samples (e.g., m columns of samples) in the block (2111) from the shared boundary (2143) and m lines of samples (e.g., m columns of samples) in the block (2112) from the shared boundary (2143).

[0265] Deblocking can be performed on one or more of the boundary regions A-D with the at least one deblocking NN, such as a deblocking NN based on DNN(s), CNN(s), or any suitable NN(s). In an example, the at least one deblocking NN includes a deblocking NN (2130). In an example, the deblocking NN (2130) is implemented using a CNN including one or more convolutional layers. The deblocking NN (2130) can include additional layer(s) described in the disclosure, such as pooling layer(s), fully connected layer(s), normalization layer(s), and/or the like. The layers in the deblocking NN (2130) can be arranged in any suitable order and in any suitable architecture (e.g., a feed-forward architecture, a recurrent architecture). In an example, a convolutional layer is followed by other layer(s), such as pooling layer(s), fully connected layer(s), normalization layer(s), and/or the like.

[0266] Deblocking can be performed on the boundary regions A-D with the deblocking NN (2130). One or more of the boundary regions A-D include artifacts. The artifacts may be induced by respective adjacent blocks. The one or more of the boundary regions A-D can be sent to the deblocking NN (2130) to reduce the artifacts. Thus, an input to the deblocking NN (2130) includes the one or more of the boundary regions A-D, and an output from the deblocking NN (2130) includes one or more of the boundary regions A-D that are deblocked.

[0267] Referring to FIG. 21B, the boundary regions A-D include artifacts induced by respective adjacent blocks. The boundary regions A-D can be sent to the deblocking NN (2130) to reduce the artifacts. An output from the deblocking NN (2130) includes the deblocked boundary regions A'-D'. In an example, artifacts in the deblocked boundary regions A'-D' are reduced compared to the artifacts in the boundary regions A-D.

[0268] Referring to FIGs. 21B and 21C, the boundary regions A-D in the image (2101) are updated, for example by being replaced by the deblocked boundary regions A'-D'. Thus, an image (2150) is generated and includes the deblocked boundary regions A'-D' and the non-boundary regions (2121)-(2124).

[0269] One or more samples can be in multiple boundary regions. When multiple boundary regions are replaced by corresponding deblocked boundary regions, any suitable method can be used to determine a value of one of the one or more shared samples.

[0270] Referring to FIG. 21A, a sample S is in the boundary regions A and C. After obtaining the boundary regions A' and C', the following methods can be used to obtain a value of the sample S. In an example, the boundary region A is replaced by the deblocked boundary region A' and subsequently, the boundary region C is replaced by the deblocked boundary region C'. Thus, the value of the sample S is determined by a value of the sample S in the deblocked boundary region C'.

[0271] In an example, the boundary region C is replaced by the deblocked boundary region C' and subsequently, the boundary region A is replaced by the deblocked boundary region A'. Thus, the value of the sample S is determined by a value of the sample S in the deblocked boundary region A'.

[0272] In an example, the value of the sample S is determined by an average (e.g., a weighted average) of the value of the sample S in the deblocked boundary region A' and the value of the sample S in the deblocked boundary region C'.

[0273] A boundary region can include samples of more than two blocks. FIG. 22 shows an example of boundary regions including samples of more than two blocks according to embodiments of the disclosure. A single boundary region AB can include the boundary regions A and B. The boundary region AB can include samples on both sides of the shared boundary (2141) between the two neighboring reconstructed blocks (2111) and (2113), and include samples on both sides of the shared boundary (2142) between the two neighboring reconstructed blocks (2112) and (2114). A single boundary region CD can include the boundary regions C and D. The boundary region CD can include samples on both sides of the shared boundary (2143) between the two neighboring reconstructed blocks (2111) and (2112), and include samples on both sides of the shared boundary (2144) between the two neighboring reconstructed blocks (2113) and (2114).

[0274] A deblocking NN, such as the deblocking NN (2130), can perform deblocking on one or more of the boundary regions AB and CD to generate one or more deblocked ones of the boundary regions. Referring to FIG. 22, the boundary regions AB and CD are sent to the deblocking NN (2130) and deblocked boundary regions AB' and CD' are generated. The deblocked boundary regions AB' and CD' can replace the boundary regions AB and CD in the image (2101), and thus an image (2250) is generated. The image (2250) can include the deblocked boundary regions AB'-CD' and the non-boundary regions (2121)-(2124).

[0275] According to embodiments of the disclosure, a multi-model deblocking method can be used. Different deblocking models can be applied to different types or categories of boundary regions, to remove artifacts. A classification module can be applied to classify the boundary regions into different categories. Any classification module can be applied. In an example, the classification module is based on NN. In an example, the classification module is not based on NN. The boundary regions can be sent to different deblocking models according to the respective categories.

[0276] In an embodiment, the at least one deblocking NN includes multiple deblocking NNs implemented based on different deblocking models, respectively. Which of the multiple deblocking NNs to apply to a boundary region can be determined. Deblocking can be performed on the boundary region with the determined deblocking NN. In an example, which of the multiple deblocking NNs to apply is determined by a classification module that is based on a NN (e.g., also referred to as a classification NN), such as a DNN, a CNN, or the like.

[0277] FIG. 23 shows an exemplary deblocking process (2300) based on multiple deblocking models according to an embodiment of the disclosure. A classification module (2310) can classify the boundary regions A-D into one or more categories. For example, the boundary regions C-D are classified into a first category, the boundary region B is classified into a second category, and the boundary region A is classified into a third category. Different deblocking models can be applied to boundary regions in different categories. In FIG. 23, a deblocking NN (2330) can be used to perform deblocking, such as a multi-model deblocking based on multiple deblocking models (e.g., deblocking models 1-L). L is a positive integer. When L is 1, the deblocking NN (2330) includes a single deblocking model. When L is larger than 1, the deblocking NN (2330) includes multiple deblocking models.

[0278] In an example, the deblocking model 1 is applied to boundary region(s) (e.g., C and D) in the first category, and deblocked boundary region(s) (e.g., C'' and D'') are generated. The deblocking model 2 is applied to boundary region(s) (e.g., B) in the second category, and deblocked boundary region(s) (e.g., B'') are generated. The deblocking model 3 is applied to boundary region(s) (e.g., A) in the third category, and deblocked boundary region(s) (e.g., A'') are generated. The deblocked boundary regions A"-D" can replace the corresponding boundary regions A-D in the image (2101), and thus an image (2350) is generated. The image (2350) can include the deblocked boundary regions A"-D" and the non-boundary regions (2121)-(2124).

[0279] Any suitable metrics can be applied to classify or categorize a boundary region. In an example, a boundary region is classified according to content of the boundary region. For example, a boundary region with a high frequency content (e.g., content with a relatively large variance) and a boundary region with a low frequency content (e.g., content with a relatively small variance) are classified into different categories corresponding to different deblocking models. Strength of artifacts in a boundary region can be used to classify the boundary region. The multi-model deblocking method can be applied to any suitable boundary regions, such as the boundary regions (e.g., A, B, C, D, AB, and/or CD) between two or more blocks. A frequency of a boundary region may be determined based on a maximum difference of samples within the boundary region. In an example, a first difference of samples near a first edge in a first side of a shared boundary is determined. In an example, a second difference of samples near a second edge in a second side of the shared boundary is determined. In an example, the first difference and the second difference are determined.

[0280] A deblocking NN (e.g., the deblocking NN (2130) in FIG. 21B or the deblocking NN (2330) in FIG. 23) can be applied to remove artifacts among blocks. In an example, samples or pixels close to a shared boundary can be deblocked more than samples (or pixels) that are further away from the shared boundary. Referring back to FIG. 21A, the sample S is closer to the shared boundary (2141) than a sample F, and thus the sample S can be deblocked more than the sample F.

[0281] A deblocking model in a deblocking NN (e.g., the deblocking NN (2130) in FIG. 21B or the deblocking NN (2330) in FIG. 23) can include one or more convolution layers. For example, a CNN-based attention mechanism (e.g., non-local attention, a Squeeze-and-Excitation Network (SENet)), a residual neural network (ResNet) (e.g., including a set of CNNs or convnets and an activation function), and/or the like may be used. For example, a DNN used by image super-resolution can be used, for example, by changing an output size to be identical to an input size. In image super-resolution, a resolution of an image can be enhanced from a low-resolution to a high-resolution.

[0282] How to perform deblocking on boundary region(s) with a NN or other learning-based methods is described above. In some examples, a video encoder and/or a video decoder may select between the deblocking method based on NN or a deblocking method not based on NN. The selection can be made on various levels, such as at a slice level, a picture level, for a group of pictures, a sequence level, and/or the like. The selection can be signaled using a flag. The selection can be inferred from content of a boundary region.

[0283] A video encoder and/or a video decoder may apply various level of boundary strength in addition to the methods and embodiments described in the disclosure, for example, when NN derived adjustments on pixels or samples are at a default level of a boundary strength (BS). By analyzing boundary conditions and block coding features, different levels of BS may be assigned to modify (e.g., enlarge or reduce) the default adjustment.

[0284] According to an embodiment of the disclosure, the at least one post-processing NN can include at least one enhancement NN. One or more of non-boundary regions of neighboring reconstructed blocks can be enhanced with the at least one enhancement NN. The one or more of the non-boundary regions can be replaced with the enhanced one or more of the non-boundary regions.

[0285] A reconstructed image (e.g., the image (2101) in FIG. 21A) can be sent to an enhancement module to generate an enhanced image (e.g., a final reconstructed image). The

reconstructed image can be sent to the enhancement module after reducing the artifacts by using a deblocking NN in some embodiments in which deblocking is performed. To enhance the quality of the image, a NN-based post-enhancement model (e.g., a post-enhancement model based on DNN(s) or CNN(s)) can be used in a post-enhancement module, such as a post-enhancement NN (2430) in FIG. 24.

[0286] FIG. 24 shows an exemplary enhancement process (2400) according to an embodiment of the disclosure. In some examples, the non-boundary regions (2121)-(2124) (e.g., remaining regions other than the boundary regions A-D) in the image (2101) are not sent to a deblocking module (e.g., the deblocking NN(2130)). In an example, a non-boundary region (e.g., the non-boundary regions (2121)) is from a reconstructed block (e.g., (2111)) in an image (e.g., (2101)), and a size of the boundary region can be (n-m)×(n-m). As described with reference to FIG. 21A, n is the side length (e.g., the width and/or the height) of the reconstructed block (e.g., (2111)), and m is a side length of the sub-boundary region (e.g., A1) for deblocking. One or more of the non-boundary regions (2121)-(2124) can be sent to the enhancement module to further increase a quality of the one or more of the non-boundary regions (2121)-(2124). The enhanced one or more of the non-boundary regions can replace the one or more of the non-boundary regions (2121)-(2124) in the image. Referring to FIG. 24, the non-boundary regions (2121)-(2124) are fed into the post-enhancement NN (2430) to generate enhanced non-boundary regions (2121')-(2124'). The enhanced non-boundary regions (2121')-(2124') can replace the enhanced non-boundary regions (2121)-(2124) to generate the enhanced image (2450).

[0287] In an example, a non-boundary region overlaps with a boundary region such that a portion of the non-boundary region is in the boundary region. In an example, the non-boundary region is the entire coding block. Referring to FIG. 24, the block (2111) can be the non-boundary region, and thus the non-boundary region (2111) borders with other neighboring blocks, such as (2112)-(2113).

[0288] In some embodiments, the at least one enhancement NN are based on multiple enhancement models (e.g., post-enhancement models), respectively. Which of the multiple enhancement models to apply to a non-boundary region can be determined, for example, by a classification module. The non-boundary region can be enhanced with the determined enhancement model. In an example, which of the multiple enhancement models to apply is determined by a classification module that is based on a NN (e.g., also referred to as a classification NN), such as a DNN, a CNN, or the like. A classification module (e.g., a

classification module (2510) used in the post-enhancement process (e.g., (2500)) can be identical to or different from a classification module (e.g., the classification module (2310) used in a deblocking process (e.g., (2300)). A classification module (e.g., a classification module (2510) used in the post-enhancement process can include NNs (e.g., DNN(s) or CNN(s)). In an example, a classification module (e.g., a classification module (2510) used in the post-enhancement process does not include a NN.

[0289] FIG. 25 shows an exemplary enhancement process (2500), such as a multi-model post-enhancement module according to an embodiment of the disclosure.

[0290] The classification module (2510) can classify the non-boundary regions (2121)-(2124) into one or more categories. For example, the non-boundary regions (2122)-(2123) are classified into a first category, and the non-boundary regions (2121) and (2124) are classified into a second category. Different enhancement models (e.g., post-enhancement models) can be applied to non-boundary regions in different categories. In FIG. 25, an enhancement NN (2530) can be used to perform enhancement, such as a multi-model enhancement based on multiple enhancement models (e.g., enhancement models 1-J). J is a positive integer. When J is 1, the enhancement NN (2530) includes a single enhancement model. When J is larger than 1, the enhancement NN (2530) includes multiple enhancement models.

[0291] In an example, the enhancement model 1 is applied to non-boundary region(s) (e.g., (2122)-(2123)) in the first category, and enhanced non-boundary region(s) (e.g., (2122")-(2123")) are generated. The enhancement model 2 is applied to non-boundary region(s) (e.g., (2121) and (2124)) in the second category, and enhanced non-boundary region(s) (e.g., (2121") and (2124")) are generated. The enhanced non-boundary regions (2121")-(2124") can replace the corresponding non-boundary regions (2121)-(2124) where an enhanced image (2550) includes the enhanced non-boundary regions (2121")-(2124") and the boundary regions A-D.

[0292] Any suitable metrics can be applied to classify or categorize a non-boundary region. In an example, a non-boundary region is classified according to content of the non-boundary region. For example, a non-boundary region with a high frequency content (e.g., content with a relatively large variance) and a non-boundary region with a low frequency content (e.g., content with a relatively small variance) are classified into different categories corresponding to different enhancement models.

[0293] An image can be enhanced at a block-level, as described with reference to FIGs. 21-25. An enhancement model (e.g., a post-enhancement model) can enhance an entire image. FIG. 26 shows an exemplary image-level enhancement process (2600) to enhance an entire image according to an embodiment of the disclosure. The image (2101) includes the non-boundary regions (2121)-(2124) and the boundary regions A-D, as describe in FIG. 21A. In an example, the image (2101) is a reconstructed image including the reconstructed blocks (2111)-(2114), as described above. Artifact in boundary regions can be reduced, and non-boundary regions can be enhanced with improved visual quality.

[0294] Referring to FIG. 26, the image (2101) including the boundary regions A-D and the non-boundary regions (2121)-(2124) can be fed into an enhancement module (2630). The enhancement module (2630) can generate enhanced boundary regions E-H corresponding to the boundary regions A-D, respectively, for example, by deblocking the boundary regions A-D. The enhancement module (2630) can generate enhanced non-boundary regions (2621)-(2624) corresponding to the non-boundary regions (2121)-(2124), respectively. The enhanced boundary regions E-H can replace the boundary regions A-D, respectively, and the enhanced non-boundary regions (2621)-(2624) can replace the non-boundary regions (2121)-(2124), respectively, and thus an enhanced image (2650) is generated based on the reconstructed image (2101).

[0295] In an example, the image based enhancement module (2630) includes an enhancement NN that can perform both deblocking and enhancement. In an example, the image based enhancement module (2630) includes an enhancement NN that can perform enhancement and a deblocking NN that can perform deblocking.

[0296] The enhancement modules (e.g., (2430), (2530), and (2630)) described with reference to FIGs. 24-26 can enhance quality of an image. The enhancement modules (e.g., (2430), (2530), and (2630)) can include one or more convolution layers. CNN-based attention mechanism (e.g., non-local attention, SENet), a ResNet (e.g., including a set of CNNs or convnets and an activation function), and/or the like may be used. For example, a DNN used by image super-resolution can be used, for example, by changing an output size to be identical to an input size.

[0297] Boundary regions and non-boundary regions in an image can be processed by an enhancement NN and a deblocking NN in any suitable order, such as sequentially or concurrently. In an example, the boundary regions are deblocked by the deblocking NN, and subsequently, the non-boundary regions are processed by the enhancement NN. In an

example, the non-boundary regions are processed by the enhancement NN, and subsequently, the boundary regions are deblocked by the deblocking NN.

[0298] According to an embodiment of the disclosure, an enhancement NN (e.g., (2430), (2530), or (2630)), a deblocking NN (e.g., (2130) or (2330)), and/or a classification NN (e.g., (2310) or (2510)) can include any neural network architecture, can include any number of layers, can include one or more sub-neural networks, as described in the disclosure, and can be trained with any suitable training images or training blocks. The training images can include raw images or images including residue data. The training blocks can be from raw images or images including residue data.

[0299] Content-adaptive online training can be applied to update one or more pretrained parameters in one of the enhancement NN (e.g., (2430), (2530), or (2630)), the deblocking NN (e.g., (2130) or (2330)), and/or the classification NN (e.g., (2310) or (2510)), as described in the disclosure.

[0300] The enhancement NN (e.g., (2430), (2530), or (2630)), the deblocking NN (e.g., (2130) or (2330)), and/or the classification NN (e.g., (2310) or (2510)) can be trained separately, for example, a single deblocking NN is trained to determine pretrained parameters in the deblocking NN. The enhancement NN (e.g., (2430), (2530), or (2630)), the deblocking NN (e.g., (2130) or (2330)), and/or the classification NN (e.g., (2310) or (2510)) can be trained (either pretrained or trained online) as a component in a NIC framework. For example, the NIC framework (900) and at least one of the enhancement NN (e.g., (2430), (2530), or (2630)), the deblocking NN (e.g., (2130) or (2330)), and/or the classification NN (e.g., (2310) or (2510)) can be trained jointly.

[0301] In an embodiment, the post-filtering NN in the post-filtering module (1910) can include one or more of the enhancement NN (e.g., (2430), (2530), or (2630)), the deblocking NN (e.g., (2130) or (2330)), and the classification NN (e.g., (2310) or (2510)). The enhancement NN (e.g., (2430), (2530), or (2630)) and the deblocking NN (e.g., (2130) or (2330)) can be applied to a reconstructed image or a reconstructed block.

[0302] The enhancement process (e.g., (2400), (2500), or (2600)) can be referred to as the post-enhancement process, for example, as the enhancement process is performed after blocks are reconstructed. For the same reason, the enhancement module (e.g., (2430), (2530), or (2630)) can be referred to as the post-enhancement module or the post-enhancement NN.

[0303] In some examples, the reconstructed image (2101) include residue data.

[0304] FIG. 27 shows a flow chart outlining an encoding process (2700) according to an embodiment of the disclosure. The process (2700) can be used in encoding an input signal (e.g., an input block or an input image). In various embodiments, the process (2700) is executed by processing circuitry, such as the processing circuitry in the terminal devices (310), (320), (330) and (340), processing circuitry that performs functions of a video encoder (e.g., (403), (603), (703), (1600A), or (1700)), or the like. In an example, the processing circuitry performs a combination of functions, such as (i) one of the video encoder (403), the video encoder (603), and the video encoder (703) and (ii) one of the video encoder (1600A) or the video encoder (1700). In some embodiments, the process (2700) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (2700). The process starts at (S2701), and proceeds to (S2710).

[0305] At (S2710), an encoded signal (e.g., $\hat{x}$) can be generated based on the input signal (e.g., x) using any suitable method. In an example, the encoded signal $\hat{x}$ is generated based on the input signal x as described with reference to FIGs. 9B and 19B. The input signal can be an input block or an input image.

[0306] At (S2720), one or more parameters of a post-filtering NN in a post-filtering module (e.g., (1910)) can be determined based on a reconstructed signal (e.g., $\bar{x}$) in the post-filtering training. Referring to FIG. 19B, the reconstructed signal $\bar{x}$ can be generated based on the encoded signal $\hat{x}$ using the NIC framework (900). The post-filtering training can be implemented to determine the one or more parameters of the post-filtering NN in the post-filtering module (1910). The post-filtering NN can be configured with initial parameters, and one or more initial parameters of the initial parameters can be iteratively updated in the post-filtering training. The one or more initial parameters can be replaced by the one or more parameters.

[0307] At (S2730), post-filtering information indicating the determined one or more parameters of the post-filtering NN can be encoded. The post-filtering information corresponds to the encoded signal $\hat{x}$. In some examples, the encoded signal $\hat{x}$ and the post-filtering information can be transmitted in the coded video bitstream. The process (2700) proceeds to (S2799), and terminates.

[0308] The process (2700) can be suitably adapted to various scenarios and steps in the process (2700) can be adjusted accordingly. One or more of the steps in the process

(2700) can be adapted, omitted, repeated, and/or combined. Any suitable order can be used to implement the process (2700). Additional step(s) can be added.

[0309] FIG. 28 shows a flow chart outlining a decoding process (2800) according to an embodiment of the disclosure. The process (2800) can be used in the reconstruction and post-filtering of an encoded signal (e.g., an encoded block or an encoded image). In various embodiments, the process (2800) is executed by processing circuitry, such as the processing circuitry in the terminal devices (310), (320), (330) and (340), the processing circuitry that performs functions of the video decoder (1600B), the processing circuitry that performs functions of a video decoder (e.g., (2000A) or (2000B)). In an example, the processing circuitry performs a combination of functions, such as (i) one of the video decoder (410), the video decoder (510), and the video decoder (810) and (ii) one of the video decoder (2000A) or the video decoder (2000B). In some embodiments, the process (2800) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (2800). The process starts at (S2801), and proceeds to (S2810).

[0310] At (S2810), first post-filtering information in a coded video bitstream corresponding to a first encoded signal can be decoded. The first encoded signal can be an encoded block or an encoded image. The first post-filtering information can indicate a first parameter (e.g., a first post-filtering parameter) of a post-filtering NN in the video decoder. The post-filtering NN can be implemented by the post-filtering module (e.g., (1910)) in the video decoder (e.g., (2000A) or (2000B)).

[0311] In an example, the first encoded signal includes a first encoded image to be reconstructed, and the first post-filtering information corresponds to the first encoded image.

[0312] In an example, the first post-filtering information further corresponds to a second encoded image to be reconstructed.

[0313] The first parameter (e.g., the first post-filtering parameter) can be a bias term or a weight coefficient in the post-filtering NN. In an example, the first post-filtering parameter is updated based on a bias term or a weight coefficient in the post-filtering NN.

[0314] The first post-filtering information can indicate one or more parameters of the post-filtering NN in various manners. In an example, the post-filtering NN is configured with initial parameters. The first post-filtering information indicates a difference between the first parameter and the one of the initial parameters, and the first parameter can be determined

according to a sum of the difference and the one of the initial parameters. In another example, the first pos-filtering information indicates the one or more parameters directly.

[0315] In an example, a number of layers in the post-filtering NN is dependent on a size (e.g., a width, a height, or an area) of the first encoded signal.

[0316] In an example, a number of layers in the post-filtering NN is dependent on a step size or a number of steps corresponding to different blocks in the one or more blocks.

[0317] In an example, an image or video comprising one or more blocks is received. The first post-filtering parameter in the image or video corresponding to the one or more blocks to be reconstructed is decoded. The first post-filtering parameter can apply to at least one of the one or more blocks. The first post-filtering parameter has been updated by a post-filtering module in a post-filtering NN that is trained based on a training dataset.

[0318] At (S2820), the post-filtering NN in the video decoder corresponding to the first encoded signal can be determined based on the first parameter (e.g., the first post-filtering parameter), for example, indicated by the first post-filtering information. For example, at least one of the initial parameters configured for the post-filtering NN is updated (e.g., replaced) with the first parameter (e.g., the first post-filtering parameter).

[0319] At (S2830), the first encoded signal can be decoded based on the determined post-filtering NN corresponding to the first encoded signal. The first encoded signal can be reconstructed to generate a reconstructed signal. The reconstructed signal can be post-filtered to generate a post-filtered signal based on the determined post-filtering NN.

[0320] In an example, the first encoded signal includes an encoded block, the encoded block can be reconstructed. The reconstructed block can be post-filtered to generate a post-filtered block based on the determined post-filtering NN.

[0321] In an example, the first encoded signal includes the first encoded image, and the first encoded image is decoded based on the determined post-filtering NN corresponding to the first encoded image.

[0322] In an example, the second encoded image is decoded based on the determined post-filtering NN corresponding to the first encoded image.

[0323] The process (2800) proceeds to (S2899), and terminates.

[0324] The process (2800) can be suitably adapted to various scenarios and steps in the process (2800) can be adjusted accordingly. One or more of the steps in the process

(2800) can be adapted, omitted, repeated, and/or combined. Any suitable order can be used to implement the process (2800). Additional step(s) can be added.

[0325] In an example, a second post-filtering parameter in the image or video corresponding to the one or more blocks is decoded. The post-filtering NN can be determined based on the second post-filtering parameter. The second post-filtering parameter can apply to a second block in the one or more blocks. The second block can be different from the at least one of the one or more blocks. The second post-filtering parameter has been updated by the post-filtering module in the post-filtering NN.

[0326] The first post-filtering parameter can be different from the second post-filtering parameter, the first post-filtering parameter is adaptive to content of a first block of the one or more blocks, the second post-filtering parameter is adaptive to content of the second block. The first block can be different from the second block.

[0327] The first post-filtering parameter can be determined based on the content of the first block of the one or more blocks, and the second post-filtering parameter can be determined based on the content of the second block.

[0328] In an example, the first post-filtering parameter corresponds to a second image to be reconstructed, and the second image is decoded based on the determined post-filtering NN.

[0329] In an example, second post-filtering information in the coded video bitstream corresponding to a second encoded signal is decoded. The second post-filtering information indicates a second parameter. The second encoded signal is different from the first encoded signal. The post-filtering NN corresponding to the first encoded signal can be updated based on the second parameter. The updated post-filtering NN corresponds to the second encoded signal and can be configured with the first parameter and the second parameter. The first parameter and the second parameter can include (i) two different weight coefficients, (ii) two different biases, or (iii) a weight coefficient and a bias of the updated post-filtering NN. The second encoded signal can be decoded based on the updated post-filtering NN corresponding to the second encoded signal.

[0330] In an example, the first post-filtering parameter is updated in (i) a single layer of the post-filtering NN, (ii) multiple layers of the post-filtering NN, or (iii) all layers of the post-filtering NN.

[0331] Embodiments in the disclosure may be used separately or combined in any order. Further, each of the methods (or embodiments), an encoder, and a decoder may be

implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium.

[0332] This disclosure does not put any restrictions on methods used for an encoder such as a neural network based encoder, a decoder such as a neural network based decoder. Neural network(s) used in an encoder, a decoder, and/or the like can be any suitable types of neural network(s), such as a DNN, a CNN, and the like.

[0333] Thus, the content-adaptive online training methods of this disclosure can accommodate different types of NIC frameworks, e.g., different types of encoding DNNs, decoding DNNs, encoding CNNs, decoding CNNs, and/or the like.

[0334] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media. For example, FIG. 29 shows a computer system (2900) suitable for implementing certain embodiments of the disclosed subject matter.

[0335] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by one or more computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0336] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[0337] The components shown in FIG. 29 for computer system (2900) are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system (2900).

[0338] Computer system (2900) may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human interface devices can also be used to capture certain media not

necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[0339] Input human interface devices may include one or more of (only one of each depicted): keyboard (2901), mouse (2902), trackpad (2903), touch-screen (2910), data-glove (not shown), joystick (2905), microphone (2906), scanner (2907), camera (2908).

[0340] Computer system (2900) may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen (2910), data-glove (not shown), or joystick (2905), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (2909), headphones (not depicted)), visual output devices (such as screens (2910) to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[0341] Computer system (2900) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (2920) with CD/DVD or the like media (2921), thumb-drive (2922), removable hard drive or solid state drive (2923), legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[0342] Those skilled in the art should also understand that term “computer readable media” as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[0343] Computer system (2900) can also include an interface (2954) to one or more communication networks (2955). Networks can for example be wireless, wireline, optical. Networks can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks include local area networks such as Ethernet, wireless LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and

terrestrial broadcast TV, vehicular and industrial to include CANBus, and so forth. Certain networks commonly require external network interface adapters that attached to certain general purpose data ports or peripheral buses (2949) (such as, for example USB ports of the computer system (2900)); others are commonly integrated into the core of the computer system (2900) by attachment to a system bus as described below (for example Ethernet interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks, computer system (2900) can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[0344] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (2940) of the computer system (2900).

[0345] The core (2940) can include one or more Central Processing Units (CPU) (2941), Graphics Processing Units (GPU) (2942), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (2943), hardware accelerators for certain tasks (2944), graphics adapters (2950), and so forth. These devices, along with Read-only memory (ROM) (2945), Random-access memory (2946), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (2947), may be connected through a system bus (2948). In some computer systems, the system bus (2948) can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus (2948), or through a peripheral bus (2949). In an example, the screen (2910) can be connected to the graphics adapter (2950). Architectures for a peripheral bus include PCI, USB, and the like.

[0346] CPUs (2941), GPUs (2942), FPGAs (2943), and accelerators (2944) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (2945) or RAM (2946). Transitional data can be also be stored in RAM (2946), whereas permanent data can be stored for example, in the internal mass storage (2947). Fast storage and retrieve to any of the memory devices can be enabled through the use of cache memory, that can be closely associated with one or more CPU (2941), GPU (2942), mass storage (2947), ROM (2945), RAM (2946), and the like.

[0347] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[0348] As an example and not by way of limitation, the computer system having architecture (2900), and specifically the core (2940) can provide functionality as a result of processor(s) (including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core (2940) that are of non-transitory nature, such as core-internal mass storage (2947) or ROM (2945). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (2940). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (2940) and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM (2946) and modifying such data structures according to the processes defined by the software. In addition or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator (2944)), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

Appendix A: Acronyms

JEM: joint exploration model

VVC: versatile video coding

BMS: benchmark set

MV: Motion Vector

HEVC: High Efficiency Video Coding

SEI: Supplementary Enhancement Information

VUI: Video Usability Information

GOPs: Groups of Pictures

TUs: Transform Units,

PUs: Prediction Units

CTUs: Coding Tree Units

CTBs: Coding Tree Blocks

PBs: Prediction Blocks

HRD: Hypothetical Reference Decoder

SNR: Signal Noise Ratio

CPUs: Central Processing Units

GPUs: Graphics Processing Units

CRT: Cathode Ray Tube

LCD: Liquid-Crystal Display

OLED: Organic Light-Emitting Diode

CD: Compact Disc

DVD: Digital Video Disc

ROM: Read-Only Memory

RAM: Random Access Memory

ASIC: Application-Specific Integrated Circuit

PLD: Programmable Logic Device

LAN: Local Area Network

GSM: Global System for Mobile communications

LTE: Long-Term Evolution

CANBus: Controller Area Network Bus

USB: Universal Serial Bus

PCI: Peripheral Component Interconnect

FPGA: Field Programmable Gate Areas

SSD: solid-state drive

IC: Integrated Circuit

CU: Coding Unit

NIC: Neural Image Compression

R-D: Rate-Distortion

E2E: End to End

ANN: Artificial Neural Network

DNN: Deep Neural Network

CNN: Convolution Neural Network

[0349] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

WHAT IS CLAIMED IS:

1. A method for video decoding in a video decoder, comprising:

receiving an image or video comprising one or more blocks;

decoding a first post-filtering parameter in the image or video corresponding to the one or more blocks to be reconstructed, the first post-filtering parameter applying to at least one of the one or more blocks, and the first post-filtering parameter having been updated by a post-filtering module in a post-filtering neural network (NN) that is trained based on a training dataset;

determining the post-filtering NN in the video decoder corresponding to the one or more blocks based on the first post-filtering parameter; and

decoding the one or more blocks based on the determined post-filtering NN corresponding to the one or more blocks;

wherein a number of layers in the post-filtering NN is dependent on a step size or a number of steps corresponding to different blocks in the one or more blocks.

2. The method of claim 1, further comprising

decoding a second post-filtering parameter in the image or video corresponding to the one or more blocks; and

determining the post-filtering NN further based on the second post-filtering parameter, wherein

the second post-filtering parameter applies to a second block in the one or more blocks,

the second block is different from the at least one of the one or more blocks, and

the second post-filtering parameter has been updated by the post-filtering module in the post-filtering NN.

3. The method of claim 1, wherein

the first post-filtering parameter corresponds to a second image to be reconstructed, and

the method further includes decoding the second image based on the determined post-filtering NN.

4. The method of claim 2, wherein the first post-filtering parameter is different from the second post-filtering parameter, the first post-filtering parameter is adaptive to content of a first block of the one or more blocks, and the second post-filtering parameter is adaptive to content of the second block.

5. The method of claim 1, wherein the first post-filtering parameter is updated based on a bias term or a weight coefficient in the post-filtering NN.

6. The method of claim 1, wherein

the post-filtering NN is configured with initial parameters, and

the determining the post-filtering NN includes updating at least one of the initial parameters with the first post-filtering parameter.

7. The method of claim 6, wherein

coding information corresponding to the one or more blocks indicates a difference between the first post-filtering parameter and the one of the initial parameters, and

the method further includes determining the first post-filtering parameter according to a sum of the difference and the one of the initial parameters.

8. The method of claim 1, wherein the first post-filtering parameter is updated in (i) a single layer of the post-filtering NN, (ii) multiple layers of the post-filtering NN, or (iii) all layers of the post-filtering NN.

9. An apparatus for video decoding, comprising:

processing circuitry configured to:

receive an image or video comprising one or more blocks;

decode a first post-filtering parameter in the image or video corresponding to the one or more blocks to be reconstructed, the first post-filtering parameter applying to at least one of the one or more blocks, and the first post-filtering parameter having been updated by a post-filtering module in a post-filtering neural network (NN) that is trained based on a training dataset;

determine the post-filtering NN in a video decoder corresponding to the one or more blocks based on the first post-filtering parameter; and

decode the one or more blocks based on the determined post-filtering NN corresponding to the one or more blocks;

wherein a number of layers in the post-filtering NN is dependent on a step size or a number of steps corresponding to different blocks in the one or more blocks.

10. The apparatus of claim 9, wherein the processing circuitry is configured to:

decode a second post-filtering parameter in the image or video corresponding to the one or more blocks; and

determine the post-filtering NN further based on the second post-filtering parameter, wherein

the second post-filtering parameter applies to a second block in the one or more blocks,

the second block is different from the at least one of the one or more blocks, and

the second post-filtering parameter has been updated by the post-filtering module in the post-filtering NN.

11. The apparatus of claim 9, wherein

the first post-filtering parameter corresponds to a second image to be reconstructed, and

the processing circuitry is configured to decode the second image based on the determined post-filtering NN.

12. The apparatus of claim 10, wherein the first post-filtering parameter is different from the second post-filtering parameter, the first post-filtering parameter is adaptive to content of a first block of the one or more blocks, and the second post-filtering parameter is adaptive to content of the second block.

13. The apparatus of claim 9, wherein the first post-filtering parameter is updated based on a bias term or a weight coefficient in the post-filtering NN.

14. The apparatus of claim 9, wherein

the post-filtering NN is configured with initial parameters, and

the processing circuitry is configured to update at least one of the initial parameters with the first post-filtering parameter.

15. The apparatus of claim 14, wherein

coding information corresponding to the one or more blocks indicates a difference between the first post-filtering parameter and the one of the initial parameters, and

the processing circuitry is configured to determine the first post-filtering parameter according to a sum of the difference and the one of the initial parameters.

16. The apparatus of claim 9, wherein the first post-filtering parameter is updated in (i) a single layer of the post-filtering NN, (ii) multiple layers of the post-filtering NN, or (iii) all layers of the post-filtering NN.

17. A non-transitory computer-readable storage medium storing a program executable by at least one processor to perform:

receiving an image or video comprising one or more blocks;

decoding a first post-filtering parameter in the image or video corresponding to the one or more blocks to be reconstructed, the first post-filtering parameter applying to at least one of the one or more blocks, and the first post-filtering parameter having been updated by a

post-filtering module in a post-filtering neural network (NN) that is trained based on a training dataset;

determining the post-filtering NN in a video decoder corresponding to the one or more blocks based on the first post-filtering parameter; and

decoding the one or more blocks based on the determined post-filtering NN corresponding to the one or more blocks;

wherein a number of layers in the post-filtering NN is dependent on a step size or a number of steps corresponding to different blocks in the one or more blocks.

18. The non-transitory computer-readable storage medium of claim 17, wherein

the program executable by the at least one processor performs:

decoding a second post-filtering parameter in the image or video corresponding to the one or more blocks, and

determining the post-filtering NN further based on the second post-filtering parameter, and

the second post-filtering parameter applies to a second block in the one or more blocks,

the second block is different from the at least one of the one or more blocks, and

the second post-filtering parameter has been updated by the post-filtering module in the post-filtering NN.

19. The non-transitory computer-readable storage medium of claim 17, wherein

the first post-filtering parameter corresponds to a second image to be reconstructed, and

the program executable by the at least one processor performs: decoding the second image based on the determined post-filtering NN.

20. The non-transitory computer-readable storage medium of claim 18, wherein the first post-filtering parameter is different from the second post-filtering parameter, the first post-filtering parameter is adaptive to content of a first block of the one or more blocks, and the second post-filtering parameter is adaptive to content of the second block.