



(51) International Patent Classification:

H04N 19/117 (2014.01) H04N 19/187 (2014.01)

(21) International Application Number:

PCT/CN2024/086465

(22) International Filing Date:

07 April 2024 (07.04.2024)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PCT/CN2023/086896

07 April 2023 (07.04.2023)

CN

PCT/CN2023/088541

15 April 2023 (15.04.2023)

CN

(71) Applicants: **DOUYIN VISION CO., LTD.** [CN/CN]; Room B-0035, 2/F, No.3 Building, No. 30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **LI, Yue**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **XU, Jizheng**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **WANG, Ye-Kui**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **LIN, Chaoyi**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **LI, Junru**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

ZHANG, Li; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(74) Agent: **SHIHUI PARTNERS**; 42/F, Tower C, Beijing Yintai Centre, No. 2 Jianguomenwai Avenue, Chaoyang District, Beijing 100022 (CN).

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

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

Published:

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

(54) Title: METHOD, APPARATUS, AND MEDIUM FOR VIDEO PROCESSING

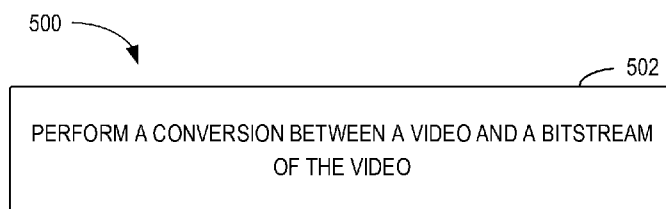


Fig. 5

(57) Abstract: Embodiments of the present disclosure provide a solution for video processing. A method for video processing is proposed. The method comprises: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture.



METHOD, APPARATUS, AND MEDIUM FOR VIDEO PROCESSING

FIELDS

[0001] Embodiments of the present disclosure relates generally to video processing techniques, and more particularly, to a neural-network post-processing filter (NNPF).

BACKGROUND

[0002] In nowadays, digital video capabilities are being applied in various aspects of peoples' lives. Multiple types of video compression technologies, such as MPEG-2, MPEG-4, ITU-TH.263, ITU-TH.264/MPEG-4 Part 10 Advanced Video Coding (AVC), ITU-TH.265 high efficiency video coding (HEVC) standard, versatile video coding (VVC) standard, have been proposed for video encoding/decoding. However, the functionality of video coding techniques is generally expected to be further improved.

SUMMARY

[0003] Embodiments of the present disclosure provide a solution for video processing.

[0004] In a first aspect, a method for video processing is proposed. The method comprises: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture.

[0005] Based on the method in accordance with the first aspect of the present disclosure, one of candidates for a purpose of NNPF indicates that decreasing a width and/or a height of the at least one picture is applicable. Compared with the conventional solution where only resolution upsampling is supported, the proposed method can advantageously make it possible for the NNPF to support resolution downsampling. Thereby, the functionality of NNPF may be extended and enhanced.

[0006] In a second aspect, an apparatus for video processing is proposed. The apparatus comprises a processor and a non-transitory memory with instructions thereon. The instructions upon execution by the processor, cause the processor to perform a method in

accordance with the first aspect of the present disclosure.

[0007] In a third aspect, a non-transitory computer-readable storage medium is proposed. The non-transitory computer-readable storage medium stores instructions that cause a processor to perform a method in accordance with the first aspect of the present disclosure.

[0008] In a fourth aspect, another non-transitory computer-readable recording medium is proposed. The non-transitory computer-readable recording medium stores a bitstream of a video which is generated by a method performed by an apparatus for video processing. The method comprises: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture.

[0009] In a fifth aspect, a method for storing a bitstream of a video is proposed. The method comprises: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture; and storing the bitstream in a non-transitory computer-readable recording medium.

[0010] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Through the following detailed description with reference to the accompanying drawings, the above and other objectives, features, and advantages of example embodiments of the present disclosure will become more apparent. In the example embodiments of the present disclosure, the same reference numerals usually refer to the

same components.

[0012] Fig. 1 illustrates a block diagram that illustrates an example video coding system, in accordance with some embodiments of the present disclosure;

[0013] Fig. 2 illustrates a block diagram that illustrates a first example video encoder,
5 in accordance with some embodiments of the present disclosure;

[0014] Fig. 3 illustrates a block diagram that illustrates an example video decoder, in accordance with some embodiments of the present disclosure;

[0015] Fig. 4 illustrates an illustration of luma data channels;

[0016] Fig. 5 illustrates a flowchart of a method for video processing in accordance with
10 embodiments of the present disclosure; and

[0017] Fig. 6 illustrates a block diagram of a computing device in which various embodiments of the present disclosure can be implemented.

[0018] Throughout the drawings, the same or similar reference numerals usually refer to the same or similar elements.

15 DETAILED DESCRIPTION

[0019] Principle of the present disclosure will now be described with reference to some embodiments. It is to be understood that these embodiments are described only for the purpose of illustration and help those skilled in the art to understand and implement the present disclosure, without suggesting any limitation as to the scope of the disclosure.

20 The disclosure described herein can be implemented in various manners other than the ones described below.

[0020] In the following description and claims, unless defined otherwise, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skills in the art to which this disclosure belongs.

25 [0021] References in the present disclosure to “one embodiment,” “an embodiment,” “an example embodiment,” and the like indicate that the embodiment described may include a particular feature, structure, or characteristic, but it is not necessary that every embodiment includes the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular
30 feature, structure, or characteristic is described in connection with an example

embodiment, it is submitted that it is within the knowledge of one skilled in the art to affect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0022] It shall be understood that although the terms “first” and “second” etc. may be used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another. For example, a first element could be termed a second element, and similarly, a second element could be termed a first element, without departing from the scope of example embodiments. As used herein, the term “and/or” includes any and all combinations of one or more of the listed terms.

[0023] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of example embodiments. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms “comprises”, “comprising”, “has”, “having”, “includes” and/or “including”, when used herein, specify the presence of stated features, elements, and/or components etc., but do not preclude the presence or addition of one or more other features, elements, components and/or combinations thereof.

Example Environment

[0024] Fig. 1 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure. As shown, the video coding system 100 may include a source device 110 and a destination device 120. The source device 110 can be also referred to as a video encoding device, and the destination device 120 can be also referred to as a video decoding device. In operation, the source device 110 can be configured to generate encoded video data and the destination device 120 can be configured to decode the encoded video data generated by the source device 110. The source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[0025] The video source 112 may include a source such as a video capture device. Examples of the video capture device include, but are not limited to, an interface to receive video data from a video content provider, a computer graphics system for generating video

data, and/or a combination thereof.

[0026] The video data may comprise one or more pictures. The video encoder 114 encodes the video data from the video source 112 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. The I/O interface 116 may include a modulator/demodulator and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via the I/O interface 116 through the network 130A. The encoded video data may also be stored onto a storage medium/server 130B for access by destination device 120.

[0027] The destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122. The I/O interface 126 may include a receiver and/or a modem. The I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/server 130B. The video decoder 124 may decode the encoded video data. The display device 122 may display the decoded video data to a user. The display device 122 may be integrated with the destination device 120, or may be external to the destination device 120 which is configured to interface with an external display device.

[0028] The video encoder 114 and the video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding (VVC) standard and other current and/or further standards.

[0029] Fig. 2 is a block diagram illustrating an example of a video encoder 200, which may be an example of the video encoder 114 in the system 100 illustrated in Fig. 1, in accordance with some embodiments of the present disclosure.

[0030] The video encoder 200 may be configured to implement any or all of the techniques of this disclosure. In the example of Fig. 2, the video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0031] In some embodiments, the video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra-prediction unit 206, a residual generation unit 207, a transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

[0032] In other examples, the video encoder 200 may include more, fewer, or different functional components. In an example, the predication unit 202 may include an intra block copy (IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[0033] Furthermore, although some components, such as the motion estimation unit 204 and the motion compensation unit 205, may be integrated, but are represented in the example of Fig. 2 separately for purposes of explanation.

[0034] The partition unit 201 may partition a picture into one or more video blocks.

The video encoder 200 and the video decoder 300 may support various video block sizes.

[0035] The mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra-coded or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some examples, the mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication signal and an intra predication signal. The mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[0036] To perform inter prediction on a current video block, the motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from buffer 213 to the current video block. The motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from the buffer 213 other than the picture associated with the current video block.

[0037] The motion estimation unit 204 and the motion compensation unit 205 may

perform different operations for a current video block, for example, depending on whether the current video block is in an I-slice, a P-slice, or a B-slice. As used herein, an “I-slice” may refer to a portion of a picture composed of macroblocks, all of which are based upon macroblocks within the same picture. Further, as used herein, in some aspects, “P-slices” and “B-slices” may refer to portions of a picture composed of macroblocks that are not dependent on macroblocks in the same picture.

[0038] In some examples, the motion estimation unit 204 may perform uni-directional prediction for the current video block, and the motion estimation unit 204 may search reference pictures of list 0 or list 1 for a reference video block for the current video block.

The motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. The motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. The motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video block indicated by the motion information of the current video block.

[0039] Alternatively, in other examples, the motion estimation unit 204 may perform bi-directional prediction for the current video block. The motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. The motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. The motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. The motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[0040] In some examples, the motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder. Alternatively, in some embodiments, the motion estimation unit 204 may signal the motion information of the current video block with reference to the motion information of another video block. For example, the

motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[0041] In one example, the motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

[0042] In another example, the motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[0043] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode signaling.

[0044] The intra prediction unit 206 may perform intra prediction on the current video block. When the intra prediction unit 206 performs intra prediction on the current video block, the intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[0045] The residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block (s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[0046] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and the residual generation unit 207 may not perform the subtracting operation.

[0047] The transform processing unit 208 may generate one or more transform

coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[0048] After the transform processing unit 208 generates a transform coefficient video block associated with the current video block, the quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[0049] The inverse quantization unit 210 and the inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. The reconstruction unit 212 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current video block for storage in the buffer 213.

[0050] After the reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed to reduce video blocking artifacts in the video block.

[0051] The entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When the entropy encoding unit 214 receives the data, the entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[0052] Fig. 3 is a block diagram illustrating an example of a video decoder 300, which may be an example of the video decoder 124 in the system 100 illustrated in Fig. 1, in accordance with some embodiments of the present disclosure.

[0053] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of Fig. 3, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0054] In the example of Fig. 3, the video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an inverse transformation unit 305, and a reconstruction unit 306

and a buffer 307. The video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200.

[0055] The entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). The entropy decoding unit 301 may decode the entropy coded video data, and from the entropy decoded video data, the motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. The motion compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode. AMVP is used, including derivation of several most probable candidates based on data from adjacent PBs and the reference picture. Motion information typically includes the horizontal and vertical motion vector displacement values, one or two reference picture indices, and, in the case of prediction regions in B slices, an identification of which reference picture list is associated with each index. As used herein, in some aspects, a “merge mode” may refer to deriving the motion information from spatially or temporally neighboring blocks.

[0056] The motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[0057] The motion compensation unit 302 may use the interpolation filters as used by the video encoder 200 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. The motion compensation unit 302 may determine the interpolation filters used by the video encoder 200 according to the received syntax information and use the interpolation filters to produce predictive blocks.

[0058] The motion compensation unit 302 may use at least part of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence. As used herein, in some aspects, a “slice” may refer to a data structure that can be decoded independently from other slices of the same picture, in terms of entropy coding, signal

prediction, and residual signal reconstruction. A slice can either be an entire picture or a region of a picture.

[0059] The intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. The
5 inverse quantization unit 304 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. The inverse transform unit 305 applies an inverse transform.

[0060] The reconstruction unit 306 may obtain the decoded blocks, e.g., by summing the residual blocks with the corresponding prediction blocks generated by the motion
10 compensation unit 302 or intra-prediction unit 303. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in the buffer 307, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

[0061] Some exemplary embodiments of the present disclosure will be described in
15 detailed hereinafter. It should be understood that section headings are used in the present document to facilitate ease of understanding and do not limit the embodiments disclosed in a section to only that section. Furthermore, while certain embodiments are described with reference to Versatile Video Coding or other specific video codecs, the disclosed
20 techniques are applicable to other video coding technologies also. Furthermore, while some embodiments describe video coding steps in detail, it will be understood that corresponding steps decoding that undo the coding will be implemented by a decoder. Furthermore, the term video processing encompasses video coding or compression, video decoding or decompression and video transcoding in which video pixels are represented
25 from one compressed format into another compressed format or at a different compressed bitrate.

1. Initial discussion

[0062] This document is related to image/video coding technologies. Specifically, this disclosure is related to the definition and signalling of purposes with downsampling capabilities
30 for neural-network post processing filters signalled in a video bitstream. Downsampling herein can be bit depth downsampling, e.g., from 10 bits to 2 bits. Downsampling herein can also be

decreasing either or both of the picture width and the picture height. Or both. The ideas may be applied individually or in various combinations, for video bitstreams coded by any codec, e.g., the versatile video coding (VVC) standard and/or the versatile supplemental enhancement information (SEI) messages for coded video bitstreams (VSEI) standard.

5 2. Abbreviations

[0063] adaptation parameter set (APS), access unit (AU), coded layer video sequence (CLVS), coded layer video sequence start (CLVSS), cyclic redundancy check (CRC), coded video sequence (CVS), finite impulse response (FIR), intra random access point (IRAP), network abstraction layer (NAL), picture parameter set (PPS), picture unit (PU), random access
10 skipped leading (RASL) picture, supplemental enhancement information (SEI), step-wise temporal sublayer access (STSA), video coding layer (VCL), versatile supplemental enhancement information as described in Rec. ITU-T H.274 | ISO/IEC 23002-7 (VSEI), video usability information (VUI), versatile video coding as described in Rec. ITU-T H.266 | ISO/IEC 23090-3 (VVC).

15 3. Further discussion

3.1 Video coding standards

[0064] Video coding standards have evolved primarily through the development of International Telecommunication Union (ITU) telecommunication standardization sector (ITU-T) and International Organization for Standardization (ISO) / International Electrotechnical
20 Commission (IEC) standards. The ITU-T produced H.261 and H.263, ISO/IEC produced motion picture experts group (MPEG)-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/ high efficiency video coding (HEVC) standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus
25 transform coding are utilized. To explore video coding technologies beyond high efficiency video coding (HEVC), the Joint Video Exploration Team (JVET) was founded by video coding experts group (VCEG) and motion picture experts group (MPEG). Further, methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). The JVET was later renamed to be the Joint Video Experts Team (JVET) when the Versatile
30 Video Coding (VVC) project officially started. VVC is a coding standard targeting at 50% bitrate reduction as compared to HEVC.

[0065] The Versatile Video Coding (VVC) standard (ITU-T H.266 | ISO/IEC 23090-3) and the associated Versatile Supplemental Enhancement Information for coded video bitstreams (VSEI) standard (ITU-T H.274 | ISO/IEC 23002-7) are designed for use in a maximally broad range of applications, including both the simple uses such as television broadcast, video conferencing, or playback from storage media, and also more advanced use cases such as adaptive bit rate streaming, video region extraction, composition and merging of content from multiple coded video bitstreams, multiview video, scalable layered coding, and viewport-adaptive 360° immersive media.

[0066] The Essential Video Coding (EVC) standard (ISO/IEC 23094-1) is another video coding standard under development by MPEG.

3.2 SEI messages in general and in VVC and VSEI

[0067] SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance. Some SEI messages are required for checking bitstream conformance and for output timing decoder conformance. Other SEI messages are not required for check bitstream conformance.

[0068] Annex D of VVC specifies syntax and semantics for SEI message payloads for some SEI messages, and specifies the use of the SEI messages and VUI parameters for which the syntax and semantics are specified in ITU-T H.274 | ISO/IEC 23002-7.

3.3 Signalling of neural-network post filters

[0069] An excerpt from the specification of two SEI messages for signalling of neural-network post-filters, as follows.

8.28 Neural-network post-filter characteristics SEI message

8.28.1 Neural-network post-filter characteristics SEI message syntax

nn_post_filter_characteristics(payloadSize) {	Descriptor
nnpfc_purpose	u(16)
nnpfc_id	ue(v)
nnpfc_mode_idc	ue(v)

if(nnpfc_mode_idc == 1) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_a	u(1)
nnpfc_tag_uri	st(v)
nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)
if(nnpfc_property_present_flag) {	
nnpfc_base_flag	u(1)
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if((nnpfc_purpose & 0x02) != 0)	
nnpfc_out_sub_c_flag	u(1)
if((nnpfc_purpose & 0x20) != 0)	
nnpfc_out_colour_format_idc	u(2)
if((nnpfc_purpose & 0x04) != 0) {	
nnpfc_pic_width_in_luma_samples	ue(v)
nnpfc_pic_height_in_luma_samples	ue(v)
}	
if((nnpfc_purpose & 0x08) != 0) {	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_output_flag[i]	u(1)
}	
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_inp_order_idc	ue(v)
nnpfc_auxiliary_inp_idc	ue(v)

nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	
nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
nnpfc_matrix_coeffs	u(8)
}	
nnpfc_out_format_idc	ue(v)
if(nnpfc_out_format_idc == 1) {	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_out_order_idc	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
nnpfc_luma_padding_val	ue(v)
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)
if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)
nnpfc_num_parameters_idc	u(6)

nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
}	
/* ISO/IEC 15938-17 bitstream */	
if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

8.28.2 Neural-network post-filter characteristics SEI message semantics

[0070] The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

[0071] Use of this SEI message requires the definition of the following variables:

- Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- Luma sample array CroppedYPic[idx] and chroma sample arrays CroppedCbPic[idx] and CroppedCrPic[idx], when present, of the input pictures with index idx in the range of 0 to numInputPics – 1, inclusive, that are used as input for the NNPF.
- Bit depth BitDepth_Y for the luma sample array of the input pictures.
- Bit depth BitDepth_C for the chroma sample arrays, if any, of the input pictures.
- A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause 7.3.
- When nnpfc_auxiliary_inp_idc is equal to 1, a filtering strength control value StrengthControlVal that shall be a real number in the range of 0 to 1, inclusive.

[0072] Input picture with index 0 corresponds to the picture for which the NNPF defined by this NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in the range of 1 to numInputPics – 1, inclusive, precedes the input picture with index i – 1 in

output order.

[0073] When `nnpfc_purpose & 0x08` is not equal to 0 and the input picture with index 0 is associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5, all input pictures are associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5 and the same value of `fp_current_frame_is_frame0_flag`.

[0074] The variables `SubWidthC` and `SubHeightC` are derived from `ChromaFormatIdc` as specified by Table 2.

[0075] NOTE 1 – More than one NNPF SEI message can be present for the same picture. When more than one NNPF SEI message with different values of `nnpfc_id` is present or activated for the same picture, they can have the same or different values of `nnpfc_purpose` and `nnpfc_mode_idc`.

[0076] `nnpfc_purpose` indicates the purpose of the NNPF as specified in Table 20.

[0077] The value of `nnpfc_purpose` shall be in the range of 0 to 63, inclusive, in bitstreams conforming to this edition of this document. Values of 64 to 65 535, inclusive, for `nnpfc_purpose` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_purpose` in the range of 64 to 65 535, inclusive.

Table 20 – Definition of `nnpfc_purpose`

Value	Interpretation
<code>nnpfc_purpose == 0</code>	May be used as determined by the application
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x01) == 0</code>	No general visual quality improvement
<code>(nnpfc_purpose & 0x01) != 0</code>	With general visual quality improvement
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x02) == 0</code>	No chroma upsampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format)
<code>(nnpfc_purpose & 0x02) != 0</code>	With chroma upsampling
<code>nnpfc_purpose > 0 &&</code>	No resolution upsampling (increasing the width or height)

(nnpfc_purpose & 0x04) == 0	
(nnpfc_purpose & 0x04) != 0	With resolution upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x08) == 0	No picture rate upsampling
(nnpfc_purpose & 0x08) != 0	With picture rate upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x10) == 0	No bit depth upsampling (increasing the luma bit depth or the chroma bit depth)
(nnpfc_purpose & 0x10) != 0	With bit depth upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x20) == 0	No colourization (from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format)
(nnpfc_purpose & 0x20) != 0	With colourization

[0078] NOTE 2– When a reserved value of nnpfc_purpose is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by nnpfc_purpose being equal to that value.

5 **[0079]** When ChromaFormatIdc is equal to 3, nnpfc_purpose & 0x02 shall be equal to 0.

[0080] When ChromaFormatIdc or nnpfc_purpose & 0x02 is not equal to 0, nnpfc_purpose & 0x20 shall be equal to 0.

[0081] nnpfc_id contains an identifying number that may be used to identify an NNPF. The value of nnpfc_id shall be in the range of 0 to 232 – 2, inclusive. Values of nnpfc_id from 256 to 511, inclusive, and from 231 to 232 – 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPF SEI message with nnpfc_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 – 2, inclusive, shall ignore the SEI message.

[0082] When an NNPF SEI message is the first NNPF SEI message, in decoding order,

that has a particular `nnpfc_id` value within the current CLVS, the following applies:

- This SEI message specifies a base NNPF.
- This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS.

5 [0083] `nnpfc_mode_idc` equal to 0 indicates that this SEI message contains an ISO/IEC 15938-17 bitstream that specifies a base NNPF or is an update relative to the base NNPF with the same `nnpfc_id` value.

[0084] When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, `nnpfc_mode_idc` equal to 1
10 specifies that the base NNPF associated with the `nnpfc_id` value is a neural network identified by the URI indicated by `nnpfc_uri` with the format identified by the tag URI `nnpfc_tag_uri`.

[0085] When an NNPF SEI message is neither the first NNPF SEI message, in decoding order, nor a repetition of the first NNPF SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, `nnpfc_mode_idc` equal to 1 specifies that an update
15 relative to the base NNPF with the same `nnpfc_id` value is defined by the URI indicated by `nnpfc_uri` with the format identified by the tag URI `nnpfc_tag_uri`.

[0086] The value of `nnpfc_mode_idc` shall be in the range of 0 to 1, inclusive, in bitstreams conforming to this edition of this document. Values of 2 to 255, inclusive, for `nnpfc_mode_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams
20 conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_mode_idc` in the range of 2 to 255, inclusive. Values of `nnpfc_mode_idc` greater than 255 shall not be present in bitstreams conforming to this edition of this document and are not reserved for future use.

[0087] When this SEI message is the first NNPF SEI message, in decoding order, that has
25 a particular `nnpfc_id` value within the current CLVS, the NNPF `PostProcessingFilter()` is assigned to be the same as the base NNPF.

[0088] When this SEI message is neither the first NNPF SEI message, in decoding order, nor a repetition of the first NNPF SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, an NNPF `PostProcessingFilter()` is obtained by applying the
30 update defined by this SEI message to the base NNPF.

[0089] Updates are not cumulative but rather each update is applied on the base NNPF, which is the NNPF specified by the first NNPF SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS.

[0090] `nnpfc_reserved_zero_bit_a` shall be equal to 0 in bitstreams conforming to this

edition of this document. Decoders shall ignore NNPFC SEI messages in which `nnpfc_reserved_zero_bit_a` is not equal to 0.

[0091] `nnpfc_tag_uri` contains a tag URI with syntax and semantics as specified in IETF RFC 4151 identifying the format and associated information about the neural network used as a base NNPF or an update relative to the base NNPF with the same `nnpfc_id` value specified by `nnpfc_uri`.

[0092] NOTE 3 – `nnpfc_tag_uri` enables uniquely identifying the format of neural network data specified by `nnpfc_uri` without needing a central registration authority.

[0093] `nnpfc_tag_uri` equal to "tag:iso.org,2023:15938-17" indicates that the neural network data identified by `nnpfc_uri` conforms to ISO/IEC 15938-17.

[0094] `nnpfc_uri` contains a URI with syntax and semantics as specified in IETF Internet Standard 66 identifying the neural network used as a base NNPF or an update relative to the base NNPF with the same `nnpfc_id` value.

[0095] `nnpfc_property_present_flag` equal to 1 specifies that syntax elements related to the filter purpose, input formatting, output formatting, and complexity are present. `nnpfc_property_present_flag` equal to 0 specifies that no syntax elements related to the filter purpose, input formatting, output formatting, and complexity are present.

[0096] When this SEI message is the first NNPFC SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, `nnpfc_property_present_flag` shall be equal to 1.

[0097] When `nnpfc_property_present_flag` is equal to 0, the values of all syntax elements that may be present only when `nnpfc_property_present_flag` is equal to 1 and for which inference values for each of them is not specified are inferred to be equal to their corresponding syntax elements, respectively, in the NNPFC SEI message that contains the base NNPF for which this SEI provides an update.

[0098] `nnpfc_base_flag` equal to 1 specifies that the SEI message specifies the base NNPF. `nnpfc_base_flag` equal to 0 specifies that the SEI message specifies an update relative to the base NNPF. When not present, the value of `nnpfc_base_flag` is inferred to be equal to 0.

[0099] The following constraints apply to the value of `nnpfc_base_flag`:

- When an NNPFC SEI message is the first NNPFC SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, the value of `nnpfc_base_flag` shall be equal to 1.
- When an NNPFC SEI message `nnpfcB` is not the first NNPFC SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS and the value

nnpfc_base_flag is equal to 1, the NNPFC SEI message shall be a repetition of the first NNPFC SEI message nnpfcA with the same nnpfc_id, in decoding order, i.e., the payload content of nnpfcB shall be the same as that of nnpfcA.

[00100] When an NNPFC SEI message is not the first NNPFC SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS and not a repetition of the first NNPFC SEI message with that particular nnpfc_id, the following applies:

- This SEI message defines an update relative to the preceding base NNPF in decoding order with the same nnpfc_id value.
- This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS or up to but excluding the decoded picture that follows the current decoded picture in output order within the current CLVS and is associated with a subsequent NNPFC SEI message, in decoding order, having that particular nnpfc_id value within the current CLVS, whichever is earlier.

[00101] When an NNPFC SEI message nnpfcCurr is not the first NNPFC SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS, is not a repetition of the first NNPFC SEI message with that particular nnpfc_id (i.e., the value of nnpfc_base_flag is equal to 0), and the value of nnpfc_property_present_flag is equal to 1, the following constraints apply:

- The value of nnpfc_purpose in the NNPFC SEI message shall be the same as the value of nnpfc_purpose in the first NNPFC SEI message, in decoding order, that has that particular nnpfc_id value within the current CLVS.
- The values of syntax elements following nnpfc_base_flag and preceding nnpfc_complexity_info_present_flag, in decoding order, in the NNPFC SEI message shall be the same as the values of corresponding syntax elements in the first NNPFC SEI message, in decoding order, that has that particular nnpfc_id value within the current CLVS.
- Either nnpfc_complexity_info_present_flag shall be equal to 0 or both nnpfc_complexity_info_present_flag shall be equal to 1 in the first NNPFC SEI message, in decoding order, that has that particular nnpfc_id value within the current CLVS (denoted as nnpfcBase below) and all the following apply:
 - nnpfc_parameter_parameter_type_idc in nnpfcCurr shall be equal to nnpfc_parameter_parameter_type_idc in nnpfcBase.
 - nnpfc_log2_parameter_bit_length_minus3 in nnpfcCurr, when present, shall be less than or equal to nnpfc_log2_parameter_bit_length_minus3 in nnpfcBase.

- If `nnpfc_num_parameters_idc` in `nnpfcBase` is equal to 0, `nnpfc_num_parameters_idc` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_num_parameters_idc` in `nnpfcBase` is greater than 0), `nnpfc_num_parameters_idc` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_num_parameters_idc` in `nnpfcBase`.
- If `nnpfc_num_kmac_operations_idc` in `nnpfcBase` is equal to 0, `nnpfc_num_kmac_operations_idc` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_num_kmac_operations_idc` in `nnpfcBase` is greater than 0), `nnpfc_num_kmac_operations_idc` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_num_kmac_operations_idc` in `nnpfcBase`.
- If `nnpfc_total_kilobyte_size` in `nnpfcBase` is equal to 0, `nnpfc_total_kilobyte_size` in `nnpfcCurr` shall be equal to 0.
- Otherwise (`nnpfc_total_kilobyte_size` in `nnpfcBase` is greater than 0), `nnpfc_total_kilobyte_size` in `nnpfcCurr` shall be greater than 0 and less than or equal to `nnpfc_total_kilobyte_size` in `nnpfcBase`.

[00102] `nnpfc_out_sub_c_flag` specifies the values of the variables `outSubWidthC` and `outSubHeightC` when `nnpfc_purpose & 0x02` or `nnpfc_purpose & 0x40` is not equal to 0. `nnpfc_out_sub_c_flag` equal to 1 specifies that `outSubWidthC` is equal to 1 and `outSubHeightC` is equal to 1. `nnpfc_out_sub_c_flag` equal to 0 specifies that `outSubWidthC` is equal to 2 and `outSubHeightC` is equal to 1. When `ChromaFormatIdc` is equal to 2, `nnpfc_purpose & 0x40` is equal to 0, and `nnpfc_out_sub_c_flag` is present, the value of `nnpfc_out_sub_c_flag` shall be equal to 1.

[00103] `nnpfc_out_colour_format_idc`, when `nnpfc_purpose & 0x20` is not equal to 0, specifies the colour format of the NNPF output and consequently the values of the variables `outSubWidthC` and `outSubHeightC`. `nnpfc_out_colour_format_idc` equal to 1 specifies that the colour format of the NNPF output is the 4:2:0 format and `outSubWidthC` and `outSubHeightC` are both equal to 2. `nnpfc_out_colour_format_idc` equal to 2 specifies that the colour format of the NNPF output is the 4:2:2 format and `outSubWidthC` is equal to 2 and `outSubHeightC` is equal to 1. `nnpfc_out_colour_format_idc` equal to 3 specifies that the colour format of the NNPF output is the 4:2:4 format and `outSubWidthC` and `outSubHeightC` are both equal to 1. The value of `nnpfc_out_colour_format_idc` shall not be equal to 0.

[00104] When `nnpfc_purpose & 0x02` and `nnpfc_purpose & 0x20` are both equal to 0, `outSubWidthC` and `outSubHeightC` are inferred to be equal to `SubWidthC` and `SubHeightC`, respectively.

[00105] `nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` specify the width and height, respectively, of the luma sample array of the picture resulting from applying the NNPF identified by `nnpfc_id` to a cropped decoded output picture. When `nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` are not present, they are inferred to be equal to `CroppedWidth` and `CroppedHeight`, respectively. The value of `nnpfc_pic_width_in_luma_samples` shall be in the range of `CroppedWidth` to `CroppedWidth * 16 - 1`, inclusive. The value of `nnpfc_pic_height_in_luma_samples` shall be in the range of `CroppedHeight` to `CroppedHeight * 16 - 1`, inclusive.

[00106] `nnpfc_num_input_pics_minus1` plus 1 specifies the number of decoded output pictures used as input for the NNPF. The value of `nnpfc_num_input_pics_minus1` shall be in the range of 0 to 63, inclusive. When `nnpfc_purpose & 0x08` is not equal to 0, the value of `nnpfc_num_input_pics_minus1` shall be greater than 0.

[00107] `nnpfc_interpolated_pics[ i ]` specifies the number of interpolated pictures generated by the NNPF between the i -th and the $(i + 1)$ -th picture used as input for the NNPF. The value of `nnpfc_interpolated_pics[ i ]` shall be in the range of 0 to 63, inclusive. The value of `nnpfc_interpolated_pics[ i ]` shall be greater than 0 for at least one i in the range of 0 to `nnpfc_num_input_pics_minus1 - 1`, inclusive.

[00108] `nnpfc_input_pic_output_flag[ i ]` equal to 1 indicates that for the i -th input picture the NNPF generates a corresponding output picture. `nnpfc_input_pic_output_flag[ i ]` equal to 0 indicates that for the i -th input picture the NNPF does not generate a corresponding output picture.

[00109] The variables `numInputPics`, specifying the number of pictures used as input for the NNPF, and `numOutputPics`, specifying the total number of pictures resulting from the NNPF, are derived as follows:

```

numInputPics = nnpfc_num_input_pics_minus1 + 1
if ( ( nnpfc_purpose & 0x08 ) != 0 ) {
    for( i = 0, numOutputPics = 0; i < numInputPics; i++ )
        if( nnpfc_input_pic_output_flag[ i ] )
            numOutputPics++
    for( i = 0; i <= numInputPics - 2; i++ )
        numOutputPics += nnpfc_interpolated_pics[ i ]
} else
    numOutputPics = 1

```

[00110] `nnpfc_component_last_flag` equal to 1 indicates that the last dimension in the input

tensor inputTensor to the NNPF and the output tensor outputTensor resulting from the NNPF is used for a current channel. nnpfc_component_last_flag equal to 0 indicates that the third dimension in the input tensor inputTensor to the NNPF and the output tensor outputTensor resulting from the NNPF is used for a current channel.

5 **[00111]** NOTE 4 – The first dimension in the input tensor and in the output tensor is used for the batch index, which is a practice in some neural network frameworks. While formulae in the semantics of this SEI message use the batch size corresponding to the batch index equal to 0, it is up to the post-processing implementation to determine the batch size used as input to the neural network inference.

10 **[00112]** NOTE 5 – For example, when nnpfc_inp_order_idc is equal to 3 and nnpfc_auxiliary_inp_idc is equal to 1, there are 7 channels in the input tensor, including four luma matrices, two chroma matrices, and one auxiliary input matrix. In this case, the process DeriveInputTensors() would derive each of these 7 channels of the input tensor one by one, and when a particular channel of these channels is processed, that channel is referred to as the
15 current channel during the process.

[00113] nnpfc_inp_format_idc indicates the method of converting a sample value of the cropped decoded output picture to an input value to the NNPF. When nnpfc_inp_format_idc is equal to 0, the input values to the NNPF are real numbers and the functions InpY() and InpC() are specified as follows:

$$20 \quad \text{InpY}(x) = x \div ((1 \ll \text{BitDepth}_Y) - 1) \quad (77)$$

$$\text{InpC}(x) = x \div ((1 \ll \text{BitDepth}_C) - 1) \quad (78)$$

[00114] When nnpfc_inp_format_idc is equal to 1, the input values to the NNPF are unsigned integer numbers and the functions InpY() and InpC() are specified as follows:

$$25 \quad \begin{aligned} &\text{shiftY} = \text{BitDepth}_Y - \text{inpTensorBitDepth}_Y \\ &\text{if}(\text{inpTensorBitDepth}_Y \geq \text{BitDepth}_Y) \\ &\quad \text{InpY}(x) = x \ll (\text{inpTensorBitDepth}_Y - \text{BitDepth}_Y) \end{aligned} \quad (79)$$

else

$$\text{InpY}(x) = \text{Clip3}(0, (1 \ll \text{inpTensorBitDepth}_Y) - 1, (x + (1 \ll (\text{shiftY} - 1)))) \gg \text{shiftY}$$

$$30 \quad \begin{aligned} &\text{shiftC} = \text{BitDepth}_C - \text{inpTensorBitDepth}_C \\ &\text{if}(\text{inpTensorBitDepth}_C \geq \text{BitDepth}_C) \\ &\quad \text{InpC}(x) = x \ll (\text{inpTensorBitDepth}_C - \text{BitDepth}_C) \end{aligned} \quad (80)$$

else

$\text{InpC}(x) = \text{Clip3}(0, (1 \ll \text{inpTensorBitDepthC}) - 1, (x + (1 \ll (\text{shiftC} - 1))) \gg \text{shiftC})$

[00115] The variable `inpTensorBitDepthY` is derived from the syntax element `nnpfc_inp_tensor_luma_bitdepth_minus8` as specified below. The variable `inpTensorBitDepthC` is derived from the syntax element `nnpfc_inp_tensor_chroma_bitdepth_minus8` as specified below.

[00116] Values of `nnpfc_inp_format_idc` greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages that contain reserved values of `nnpfc_inp_format_idc`.

[00117] `nnpfc_inp_tensor_luma_bitdepth_minus8` plus 8 specifies the bit depth of luma sample values in the input integer tensor. The value of `inpTensorBitDepthY` is derived as follows:

$$\text{inpTensorBitDepthY} = \text{nnpfc_inp_tensor_luma_bitdepth_minus8} + 8 \quad (81)$$

[00118] It is a requirement of bitstream conformance that the value of `nnpfc_inp_tensor_luma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

[00119] `nnpfc_inp_tensor_chroma_bitdepth_minus8` plus 8 specifies the bit depth of chroma sample values in the input integer tensor. The value of `inpTensorBitDepthC` is derived as follows:

$$\text{inpTensorBitDepthC} = \text{nnpfc_inp_tensor_chroma_bitdepth_minus8} + 8 \quad (82)$$

[00120] It is a requirement of bitstream conformance that the value of `nnpfc_inp_tensor_chroma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

[00121] `nnpfc_inp_order_idc` indicates the method of ordering the sample arrays of a cropped decoded output picture as one of the input pictures to the NNPF.

[00122] The value of `nnpfc_inp_order_idc` shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this edition of this document. Values of 4 to 255, inclusive, for `nnpfc_inp_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_inp_order_idc` in the range of 4 to 255, inclusive. Values of `nnpfc_inp_order_idc` greater than 255 shall not be present in bitstreams conforming to this edition of this document and are not reserved for future use.

[00123] When ChromaFormatIdc is not equal to 1, nnpfc_inp_order_idc shall not be equal to 3.

[00124] Table 21 contains an informative description of nnpfc_inp_order_idc values.

Table 21 – Description of nnpfc_inp_order_idc values

nnpfc_inp_order_idc	Description
0	If nnpfc_auxiliary_inp_idc is equal to 0, one luma matrix is present in the input tensor for each input picture, and the number of channels is 1. Otherwise when nnpfc_auxiliary_inp_idc is equal to 1, one luma matrix and one auxiliary input matrix are present, and the number of channels is 2.
1	If nnpfc_auxiliary_inp_idc is equal to 0, two chroma matrices are present in the input tensor, and the number of channels is 2. Otherwise when nnpfc_auxiliary_inp_idc is equal to 1, two chroma matrices and one auxiliary input matrix are present, and the number of channels is 3.
2	If nnpfc_auxiliary_inp_idc is equal to 0, one luma and two chroma matrices are present in the input tensor, and the number of channels is 3. Otherwise when nnpfc_auxiliary_inp_idc is equal to 1, one luma matrix, two chroma matrices and one auxiliary input matrix are present, and the number of channels is 4.
3	If nnpfc_auxiliary_inp_idc is equal to 0, four luma matrices and two chroma matrices are present in the input tensor, and the number of channels is 6. Otherwise when nnpfc_auxiliary_inp_idc is equal to 1, four luma matrices, two chroma matrices, and one auxiliary input matrix are present in the input tensor, and the number of channels is 7. The luma channels are derived in an interleaved manner as illustrated in Fig. 4. This nnpfc_inp_order_idc can only be used when the input chroma format is 4:2:0.
4..255	Reserved

5

[00125] Fig. 4 illustrates an example of deriving the four luma channels (right) from the luma component (left) when nnpfc_inp_order_idc is equal to 3.

[00126] A patch is a rectangular array of samples from a component (e.g., a luma or chroma component) of a picture.

10 [00127] nnpfc_auxiliary_inp_idc greater than 0 indicates that auxiliary input data is present

in the input tensor of the NNPF. `nnpfc_auxiliary_inp_idc` equal to 0 indicates that auxiliary input data is not present in the input tensor. `nnpfc_auxiliary_inp_idc` equal to 1 specifies that auxiliary input data is derived as specified in Formula 84.

[00128] The value of `nnpfc_auxiliary_inp_idc` shall be in the range of 0 to 1, inclusive, in bitstreams conforming to this edition of this document. Values of 2 to 255, inclusive, for `nnpfc_inp_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_inp_order_idc` in the range of 2 to 255, inclusive. Values of `nnpfc_inp_order_idc` greater than 255 shall not be present in bitstreams conforming to this edition of this document and are not reserved for future use.

[00129] When `nnpfc_auxiliary_inp_idc` is equal to 1, the variable `strengthControlScaledVal` is derived as follows:

```

if( nnpfc_inp_format_idc == 1 )
    strengthControlScaledVal = Floor ( StrengthControlVal *
    ( ( 1 << inpTensorBitDepthY ) - 1 ) ) (83)
else
    strengthControlScaledVal = StrengthControlVal

```

[00130] The process `DeriveInputTensors()`, for deriving the input tensor `inputTensor` for a given vertical sample coordinate `cTop` and a horizontal sample coordinate `cLeft` specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

for( i = 0; i < numInputPics; i++ ) {
    if( nnpfc_inp_order_idc == 0 )
        for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
            for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++ ) {
                inpVal = InpY( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight,
                    CroppedWidth, CroppedYPic[ i ] ) )
                yPovlp = yP + nnpfc_overlap
                xPovlp = xP + nnpfc_overlap
                if( !nnpfc_component_last_flag )
                    inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpVal
                else
                    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpVal
            }
        if( nnpfc_auxiliary_inp_idc == 1 )

```

```

        if( !nnpfc_component_last_flag )
            inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] =
strengthControlScaledVal
        else
5            inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] =
strengthControlScaledVal
        }
    else if( nnpfc_inp_order_idc == 1 ) (84)
        for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
10        for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++ ) {
            inpCbVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight
/ SubHeightC,
                CroppedWidth / SubWidthC, CroppedCbPic[ i ] ) )
            inpCrVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight
15 / SubHeightC,
                CroppedWidth / SubWidthC, CroppedCrPic[ i ] ) )
            yPovlp = yP + nnpfc_overlap
            xPovlp = xP + nnpfc_overlap
            if( !nnpfc_component_last_flag ) {
20                inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpCbVal
                inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpCrVal
            } else {
                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpCbVal
                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpCrVal
25            }
        }
        if( nnpfc_auxiliary_inp_idc == 1 )
            if( !nnpfc_component_last_flag )
                inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] =
strengthControlScaledVal
30            else
                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] =
strengthControlScaledVal
            }
    else if( nnpfc_inp_order_idc == 2 )

```

```

for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
  for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++ ) {
    yY = cTop + yP
    xY = cLeft + xP
5    yC = yY / SubHeightC
    xC = xY / SubWidthC
    inpYVal = InpY( InpSampleVal( yY, xY, CroppedHeight,
                                CroppedWidth, CroppedYPic[ i ] ) )
    inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,
10    CroppedWidth / SubWidthC, CroppedCbPic[ i ] ) )
    inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,
                                CroppedWidth / SubWidthC, CroppedCrPic[ i ] ) )
    yPovlp = yP + nnpfc_overlap
    xPovlp = xP + nnpfc_overlap
15    if( !nnpfc_component_last_flag ) {
        inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpYVal
        inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpCbVal
        inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = inpCrVal
    } else {
20        inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpYVal
        inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpCbVal
        inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] = inpCrVal
    }
    if( nnpfc_auxiliary_inp_idc == 1 )
25        if( !nnpfc_component_last_flag )
            inputTensor[ 0 ][ i ][ 3 ][ yPovlp ][ xPovlp ] =
strengthControlScaledVal
            else
30                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 3 ] =
strengthControlScaledVal
        }
    else if( nnpfc_inp_order_idc == 3 )
        for( yP = -nnpfc_overlap; yP < inpPatchHeight + nnpfc_overlap; yP++)
            for( xP = -nnpfc_overlap; xP < inpPatchWidth + nnpfc_overlap; xP++ ) {

```

```

yTL = cTop + yP * 2
xTL = cLeft + xP * 2
yBR = yTL + 1
xBR = xTL + 1
5   yC = cTop / 2 + yP
xC = cLeft / 2 + xP
inpTLVal = InpY( InpSampleVal( yTL, xTL, CroppedHeight,
                              CroppedWidth, CroppedYPic[ i ] ) )
inpTRVal = InpY( InpSampleVal( yTL, xBR, CroppedHeight,
10  CroppedWidth, CroppedYPic[ i ] ) )
inpBLVal = InpY( InpSampleVal( yBR, xTL, CroppedHeight,
                              CroppedWidth, CroppedYPic[ i ] ) )
inpBRVal = InpY( InpSampleVal( yBR, xBR, CroppedHeight,
                              CroppedWidth, CroppedYPic[ i ] ) )
15  inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,
                              CroppedWidth / 2, CroppedCbPic[ i ] ) )
inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,
                              CroppedWidth / 2, CroppedCrPic[ i ] ) )
yPovlp = yP + nnpfc_overlap
20  xPovlp = xP + nnpfc_overlap
if( !nnpfc_component_last_flag ) {
    inputTensor[ 0 ][ i ][ 0 ][ yPovlp ][ xPovlp ] = inpTLVal
    inputTensor[ 0 ][ i ][ 1 ][ yPovlp ][ xPovlp ] = inpTRVal
    inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = inpBLVal
25  inputTensor[ 0 ][ i ][ 3 ][ yPovlp ][ xPovlp ] = inpBRVal
    inputTensor[ 0 ][ i ][ 4 ][ yPovlp ][ xPovlp ] = inpCbVal
    inputTensor[ 0 ][ i ][ 5 ][ yPovlp ][ xPovlp ] = inpCrVal
} else {
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 0 ] = inpTLVal
30  inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = inpTRVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 2 ] = inpBLVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 3 ] = inpBRVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 4 ] = inpCbVal
    inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 5 ] = inpCrVal

```

```

        }
        if( nnpfc_auxiliary_inp_idc == 1 )
            if( !nnpfc_component_last_flag )
                inputTensor[ 0 ][ i ][ 6 ][ yPovlp ][ xPovlp ] =
5      strengthControlScaledVal
            else
                inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 6 ] =
                strengthControlScaledVal
            }
10    }

```

[00131] nnpfc_separate_colour_description_present_flag equal to 1 indicates that a distinct combination of colour primaries, transfer characteristics, and matrix coefficients for the picture resulting from the NNPF is specified in the SEI message syntax structure. nnpfc_separate_colour_description_present_flag equal to 0 indicates that the combination of

15 colour primaries, transfer characteristics, and matrix coefficients for the picture resulting from the NNPF is the same as indicated in VUI parameters for the CLVS.

[00132] nnpfc_colour_primaries has the same semantics as specified in subclause 7.3 for the vui_colour_primaries syntax element, except as follows:

- nnpfc_colour_primaries specifies the colour primaries of the picture resulting from
- 20 applying the NNPF specified in the SEI message, rather than the colour primaries used for the CLVS.
- When nnpfc_colour_primaries is not present in the NNPF SEI message, the value of nnpfc_colour_primaries is inferred to be equal to vui_colour_primaries.

[00133] nnpfc_transfer_characteristics has the same semantics as specified in subclause 7.3

25 for the vui_transfer_characteristics syntax element, except as follows:

- nnpfc_transfer_characteristics specifies the transfer characteristics of the picture resulting from applying the NNPF specified in the SEI message, rather than the transfer characteristics used for the CLVS.
- When nnpfc_transfer_characteristics is not present in the NNPF SEI message, the value
- 30 of nnpfc_transfer_characteristics is inferred to be equal to vui_transfer_characteristics.

[00134] nnpfc_matrix_coeffs has the same semantics as specified in subclause 7.3 for the vui_matrix_coeffs syntax element, except as follows:

- nnpfc_matrix_coeffs specifies the matrix coefficients of the picture resulting from applying the NNPF specified in the SEI message, rather than the matrix coefficients used for the

CLVS.

- When `nnpfc_matrix_coeffs` is not present in the NNPFC SEI message, the value of `nnpfc_matrix_coeffs` is inferred to be equal to `vui_matrix_coeffs`.
- The values allowed for `nnpfc_matrix_coeffs` are not constrained by the chroma format of the decoded video pictures that is indicated by the value of `ChromaFormatIdc` for the semantics of the VUI parameters.
- When `nnpfc_matrix_coeffs` is equal to 0, `nnpfc_out_order_idc` shall not be equal to 1 or 3.

[00135] `nnpfc_out_format_idc` equal to 0 indicates that the sample values output by the NNPFC are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth `bitDepth` for subsequent post-processing or displaying.

[00136] `nnpfc_out_format_idc` equal to 1 indicates that the luma sample values output by the NNPFC are unsigned integer numbers in the range of 0 to $(1 \ll (\text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8)) - 1$, inclusive, and the chroma sample values output by the NNPFC are unsigned integer numbers in the range of 0 to $(1 \ll (\text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8)) - 1$, inclusive.

[00137] Values of `nnpfc_out_format_idc` greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFC SEI messages that contain reserved values of `nnpfc_out_format_idc`.

[00138] `nnpfc_out_tensor_luma_bitdepth_minus8` plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of `nnpfc_out_tensor_luma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

[00139] `nnpfc_out_tensor_chroma_bitdepth_minus8` plus 8 specifies the bit depth of chroma sample values in the output integer tensor. The value of `nnpfc_out_tensor_chroma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

[00140] When `nnpfc_purpose & 0x10` is not equal to 0, the value of `nnpfc_out_format_idc` shall be equal to 1 and at least one of the following conditions shall be true:

- `nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is greater than `BitDepthY`.
- `nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is greater than `BitDepthC`.

[00141] `nnpfc_out_order_idc` indicates the output order of samples resulting from the NNPFC.

[00142] The value of `nnpfc_out_order_idc` shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this edition of this document. Values of 4 to 255, inclusive, for `nnpfc_out_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present

in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFC SEI messages with `nnpfc_out_order_idc` in the range of 4 to 255, inclusive. Values of `nnpfc_out_order_idc` greater than 255 shall not be present in bitstreams conforming to this edition of this document and are not reserved for future use.

- 5 **[00143]** When `nnpfc_purpose & 0x02` is not equal to 0, `nnpfc_out_order_idc` shall not be equal to 3.

[00144] Table 22 contains an informative description of `nnpfc_out_order_idc` values.

Table 22 – Description of `nnpfc_out_order_idc` values

<code>nnpfc_out_order_idc</code>	Description
0	Only the luma matrix is present in the output tensor, thus the number of channels is 1.
1	Only the chroma matrices are present in the output tensor, thus the number of channels is 2.
2	The luma and chroma matrices are present in the output tensor, thus the number of channels is 3.
3	Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This <code>nnpfc_out_order_idc</code> can only be used when the output chroma format is 4:2:0.
4..255	Reserved

- 10 **[00145]** The process `StoreOutputTensors( )`, for deriving sample values in the filtered output sample arrays `FilteredYPic`, `FilteredCbPic`, and `FilteredCrPic` from the output tensor `outputTensor` for a given vertical sample coordinate `cTop` and a horizontal sample coordinate `cLeft` specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

15   for( i = 0; i < numOutputPics; i++ ) {
        if( nnpfc_out_order_idc == 0 )
            for( yP = 0; yP < outPatchHeight; yP++ )
                for( xP = 0; xP < outPatchWidth; xP++ ) {
                    yY = cTop * outPatchHeight / inpPatchHeight + yP
20                    xY = cLeft * outPatchWidth / inpPatchWidth + xP
                    if ( yY < nnpfc_pic_height_in_luma_samples && xY <

```

```

nnpfc_pic_width_in_luma_samples )
    if( !nnpfc_component_last_flag )
        FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
    else
5         FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
    }
else if( nnpfc_out_order_idc == 1 ) (85)
    for( yP = 0; yP < outPatchCHeight; yP++ )
        for( xP = 0; xP < outPatchCWidth; xP++ ) {
10         xSrc = cLeft * horCScaling + xP
        ySrc = cTop * verCScaling + yP
        if ( ySrc < nnpfc_pic_height_in_luma_samples / outSubHeightC &&
            xSrc < nnpfc_pic_width_in_luma_samples / outSubWidthC )
            if( !nnpfc_component_last_flag ) {
15         FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
            FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
            } else {
                FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
                FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
20         }
            }
        else if( nnpfc_out_order_idc == 2 )
            for( yP = 0; yP < outPatchHeight; yP++ )
                for( xP = 0; xP < outPatchWidth; xP++ ) {
25         yY = cTop * outPatchHeight / inpPatchHeight + yP
        xY = cLeft * outPatchWidth / inpPatchWidth + xP
        yC = yY / outSubHeightC
        xC = xY / outSubWidthC
        yPc = ( yP / outSubHeightC ) * outSubHeightC
30         xPc = ( xP / outSubWidthC ) * outSubWidthC
        if ( yY < nnpfc_pic_height_in_luma_samples && xY <
nnpfc_pic_width_in_luma_samples )
            if( !nnpfc_component_last_flag ) {
                FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]

```

```

        FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 1 ][ yPc ][ xPc ]
        FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 2 ][ yPc ][ xPc ]
    } else {
        FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
5      FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 1 ]
        FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 2 ]
    }
}

else if( nnpfc_out_order_idc == 3 )
10   for( yP = 0; yP < outPatchHeight; yP++ )
        for( xP = 0; xP < outPatchWidth; xP++ ) {
            ySrc = cTop / 2 * outPatchHeight / inpPatchHeight + yP
            xSrc = cLeft / 2 * outPatchWidth / inpPatchWidth + xP
            if ( ySrc < nnpfc_pic_height_in_luma_samples / 2 &&
15             xSrc < nnpfc_pic_width_in_luma_samples / 2 )
                if( !nnpfc_component_last_flag ) {
                    FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
                    FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
20   outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
                    FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ 2 ][ yP ][ xP ]
                    FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ 3 ][ yP ][ xP ]
25             FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 4 ][ yP ][ xP ]
                    FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 5 ][ yP ][ xP ]
                } else {
                    FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
30             FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
                    FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 2 ]
                    FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =

```

```

outputTensor[ 0 ][ i ][ yP ][ xP ][ 3 ]
    FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 4 ]
    FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 5 ]
    }
5      }
    }

```

[00146] nnpfc_overlap indicates the overlapping horizontal and vertical sample counts of adjacent input tensors of the NNPF. The value of nnpfc_overlap shall be in the range of 0 to 16 383, inclusive.

10 **[00147]** nnpfc_constant_patch_size_flag equal to 1 indicates that the NNPF accepts exactly the patch size indicated by nnpfc_patch_width_minus1 and nnpfc_patch_height_minus1 as input. nnpfc_constant_patch_size_flag equal to 0 indicates that the NNPF accepts as input any patch size with width inpPatchWidth and height inpPatchHeight such that the width of an extended patch (i.e., a patch plus the overlapping area), which is equal to
15 inpPatchWidth + 2 * nnpfc_overlap, is a positive integer multiple of nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap, and the height of the extended patch, which is equal to inpPatchHeight + 2 * nnpfc_overlap, is a positive integer multiple of nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap.

[00148] nnpfc_patch_width_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to
20 1, indicates the horizontal sample counts of the patch size required for the input to the NNPF. The value of nnpfc_patch_width_minus1 shall be in the range of 0 to Min(32 766, CroppedWidth - 1), inclusive.

[00149] nnpfc_patch_height_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to 1, indicates the vertical sample counts of the patch size required for the input to the NNPF. The
25 value of nnpfc_patch_height_minus1 shall be in the range of 0 to Min(32 766, CroppedHeight - 1), inclusive.

[00150] nnpfc_extended_patch_width_cd_delta_minus1 plus 1 plus 2 * nnpfc_overlap, when nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the width of an extended patch required for the input to the NNPF. The value of
30 nnpfc_extended_patch_width_cd_delta_minus1 shall be in the range of 0 to Min(32 766, CroppedWidth - 1), inclusive.

[00151] nnpfc_extended_patch_height_cd_delta_minus1 plus 1 plus 2 * nnpfc_overlap, when nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the height of an extended patch required for the input to the NNPF. The value of

nnpfc_extended_patch_height_cd_delta_minus1 shall be in the range of 0 to $\text{Min}(32766, \text{CroppedHeight} - 1)$, inclusive.

[00152] Let the variables `inpPatchWidth` and `inpPatchHeight` be the patch size width and the patch size height, respectively.

5 **[00153]** If `nnpfc_constant_patch_size_flag` is equal to 0, the following applies:

- The values of `inpPatchWidth` and `inpPatchHeight` are either provided by external means not specified in this document or set by the post-processor itself.
- The value of `inpPatchWidth + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchWidth` shall be less than or equal to `CroppedWidth`. The value of `inpPatchHeight + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchHeight` shall be less than or equal to `CroppedHeight`.

10 **[00154]** Otherwise (`nnpfc_constant_patch_size_flag` is equal to 1), the value of `inpPatchWidth` is set equal to `nnpfc_patch_width_minus1 + 1` and the value of `inpPatchHeight` is set equal to `nnpfc_patch_height_minus1 + 1`.

[00155] The variables `outPatchWidth`, `outPatchHeight`, `horCScaling`, `verCScaling`, `outPatchCWidth`, and `outPatchCHeight` are derived as follows:

$$20 \quad \text{outPatchWidth} = (\text{nnpfc_pic_width_in_luma_samples} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (86)$$

$$\text{outPatchHeight} = (\text{nnpfc_pic_height_in_luma_samples} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (87)$$

$$\text{horCScaling} = \text{SubWidthC} / \text{outSubWidthC} \quad (88)$$

$$\text{verCScaling} = \text{SubHeightC} / \text{outSubHeightC} \quad (89)$$

$$25 \quad \text{outPatchCWidth} = \text{outPatchWidth} * \text{horCScaling} \quad (90)$$

$$\text{outPatchCHeight} = \text{outPatchHeight} * \text{verCScaling} \quad (91)$$

[00156] It is a requirement of bitstream conformance that `outPatchWidth * CroppedWidth` shall be equal to `nnpfc_pic_width_in_luma_samples * inpPatchWidth` and `outPatchHeight * CroppedHeight` shall be equal to `nnpfc_pic_height_in_luma_samples * inpPatchHeight`.

30 **[00157]** `nnpfc_padding_type` indicates the process of padding when referencing sample locations outside the boundaries of the cropped decoded output picture as described in Table 23. The value of `nnpfc_padding_type` shall be in the range of 0 to 15, inclusive.

Table 23 – Informative description of `nnpfc_padding_type` values

nnpfc_padding_type	Description
0	zero padding
1	replication padding
2	reflection padding
3	wrap-around padding
4	fixed padding
5..15	Reserved

[00158] nnpfc_luma_padding_val indicates the luma value to be used for padding when nnpfc_padding_type is equal to 4.

[00159] nnpfc_cb_padding_val indicates the Cb value to be used for padding when
5 nnpfc_padding_type is equal to 4.

[00160] nnpfc_cr_padding_val indicates the Cr value to be used for padding when nnpfc_padding_type is equal to 4.

[00161] The function InpSampleVal(y, x, picHeight, picWidth, croppedPic) with inputs being a vertical sample location y, a horizontal sample location x, a picture height picHeight, a
10 picture width picWidth, and sample array croppedPic returns the value of sampleVal derived as follows:

[00162] NOTE 6 – For the inputs to the function InpSampleVal(), the vertical location is listed before the horizontal location for compatibility with input tensor conventions of some inference engines.

```

15     if( nnpfc_padding_type == 0 )
        if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
            sampleVal = 0
        else
            sampleVal = croppedPic[ x ][ y ]                                     (92)
20     else if( nnpfc_padding_type == 1 )
        sampleVal = croppedPic[ Clip3( 0, picWidth - 1, x ) ][ Clip3( 0, picHeight - 1, y ) ]
    else if( nnpfc_padding_type == 2 )
        sampleVal = croppedPic[ Reflect( picWidth - 1, x ) ][ Reflect( picHeight - 1, y ) ]
    else if( nnpfc_padding_type == 3 )
25     if( y >= 0 && y < picHeight )
        sampleVal = croppedPic[ Wrap( picWidth - 1, x ) ][ y ]

```

```

else if( nnpfc_padding_type == 4 )
    if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
        sampleVal[ 0 ] = nnpfc_luma_padding_val
        sampleVal[ 1 ] = nnpfc_cb_padding_val
5      sampleVal[ 2 ] = nnpfc_cr_padding_val
    else
        sampleVal = croppedPic[ x ][ y ]

```

[00163] The following example process may be used, with the NNPF PostProcessingFilter(), to generate, in a patch-wise manner, the filtered and/or interpolated picture(s), which contain Y, Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as indicated by nnpfc_out_order_idc:

```

if( nnpfc_inp_order_idc == 0 || nnpfc_inp_order_idc == 2 )
    for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight )
        for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth ) {
15      DeriveInputTensors( )
            outputTensor = PostProcessingFilter( inputTensor )
            StoreOutputTensors( )
        }
    else if( nnpfc_inp_order_idc == 1 )
20      for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop += inpPatchHeight )
        for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft += inpPatchWidth ) {
                                                    (93)
            DeriveInputTensors( )
            outputTensor = PostProcessingFilter( inputTensor )
25      StoreOutputTensors( )
        }
    else if( nnpfc_inp_order_idc == 3 )
        for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight * 2 )
            for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth * 2 ) {
30      DeriveInputTensors( )
            outputTensor = PostProcessingFilter( inputTensor )
            StoreOutputTensors( )
        }

```

[00164] The order of the pictures in the stored output tensor is in output order, and the output

order generated by applying the NNPF in output order is interpreted to be in output order (and not conflicting with the output order of the input pictures).

[00165] `nnpfc_complexity_info_present_flag` equal to 1 specifies that one or more syntax elements that indicate the complexity of the NNPF associated with the `nnpfc_id` are present.

5 `nnpfc_complexity_info_present_flag` equal to 0 specifies that no syntax elements that indicates the complexity of the NNPF associated with the `nnpfc_id` are present.

[00166] `nnpfc_parameter_type_idc` equal to 0 indicates that the neural network uses only integer parameters. `nnpfc_parameter_type_flag` equal to 1 indicates that the neural network may use floating point or integer parameters. `nnpfc_parameter_type_idc` equal to 2 indicates that the
10 neural network uses only binary parameters. `nnpfc_parameter_type_idc` equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_parameter_type_idc` equal to 3.

[00167] `nnpfc_log2_parameter_bit_length_minus3` equal to 0, 1, 2, and 3 indicates that the
15 neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively. When `nnpfc_parameter_type_idc` is present and `nnpfc_log2_parameter_bit_length_minus3` is not present the neural network does not use parameters of bit length greater than 1.

[00168] `nnpfc_num_parameters_idc` indicates the maximum number of neural network parameters for the NNPF in units of a power of 2 048. `nnpfc_num_parameters_idc` equal to 0
20 indicates that the maximum number of neural network parameters is unknown. The value `nnpfc_num_parameters_idc` shall be in the range of 0 to 52, inclusive. Values of `nnpfc_num_parameters_idc` greater than 52 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with
25 `nnpfc_num_parameters_idc` greater than 52.

[00169] If the value of `nnpfc_num_parameters_idc` is greater than zero, the variable `maxNumParameters` is derived as follows:

$$\text{maxNumParameters} = (2\,048 \ll \text{nnpfc_num_parameters_idc}) - 1 \quad (94)$$

[00170] It is a requirement of bitstream conformance that the number of neural network
30 parameters of the NNPF shall be less than or equal to `maxNumParameters`.

[00171] `nnpfc_num_kmac_operations_idc` greater than 0 indicates that the maximum number of multiply-accumulate operations per sample of the NNPF is less than or equal to `nnpfc_num_kmac_operations_idc * 1\,000`. `nnpfc_num_kmac_operations_idc` equal to 0

indicates that the maximum number of multiply-accumulate operations of the network is unknown. The value of `nnpfc_num_kmac_operations_idc` shall be in the range of 0 to $232 - 2$, inclusive.

[00172] `nnpfc_total_kilobyte_size` greater than 0 indicates a total size in kilobytes required to store the uncompressed parameters for the neural network. The total size in bits is a number equal to or greater than the sum of bits used to store each parameter. `nnpfc_total_kilobyte_size` is the total size in bits divided by 8 000, rounded up. `nnpfc_total_kilobyte_size` equal to 0 indicates that the total size required to store the parameters for the neural network is unknown. The value of `nnpfc_total_kilobyte_size` shall be in the range of 0 to $232 - 2$, inclusive.

[00173] `nnpfc_reserved_zero_bit_b` shall be equal to 0 in bitstreams conforming to this edition of this document. Decoders shall ignore NNPFC SEI messages in which `nnpfc_reserved_zero_bit_b` is not equal to 0.

[00174] `nnpfc_payload_byte[i]` contains the i -th byte of a bitstream conforming to ISO/IEC 15938-17. The byte sequence `nnpfc_payload_byte[i]` for all present values of i shall be a complete bitstream that conforms to ISO/IEC 15938-17.

8.29 Neural-network post-filter activation SEI message

8.29.1 Neural-network post-filter activation SEI message syntax

nn_post_filter_activation(payloadSize) {	Descriptor
nnpfa_target_id	ue(v)
nnpfa_cancel_flag	u(1)
if(!nnpfa_cancel_flag)	
nnpfa_persistence_flag	u(1)
}	

8.29.2 Neural-network post-filter activation SEI message semantics

[00175] The neural-network post-filter activation (NNPFA) SEI message activates or deactivates the possible use of the target neural-network post-processing filter (NNPF), identified by `nnpfa_target_id`, for post-processing filtering of a set of pictures. For a particular picture for which the NNPF is activated, the target NNPF is the NNPF specified by the last NNPFC SEI

message with `nnpfc_id` equal to `nnpfa_target_id`, that precedes the first VCL NAL unit of the current picture in decoding order that is not a repetition of the NNPFC SEI message that contains the base NNPF.

[00176] NOTE 1 – There can be several NNPFA SEI messages present for the same picture, for example, when the NNPFs are meant for different purposes or for filtering of different colour components.

[00177] `nnpfa_target_id` indicates the target NNPF, which is specified by one or more NNPFC SEI messages that pertain to the current picture and have `nnpfc_id` equal to `nnpfa_target_id`.

[00178] The value of `nnpfa_target_id` shall be in the range of 0 to $232 - 2$, inclusive. Values of `nnpfa_target_id` from 256 to 511, inclusive, and from 231 to $232 - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPFA SEI message with `nnpfa_target_id` in the range of 256 to 511, inclusive, or in the range of 231 to $232 - 2$, inclusive, shall ignore the SEI message.

[00179] An NNPFA SEI message with a particular value of `nnpfa_target_id` shall not be present in a current PU unless one or both of the following conditions are true:

- Within the current CLVS there is an NNPFC SEI message with `nnpfc_id` equal to the particular value of `nnpfa_target_id` present in a PU preceding the current PU in decoding order.
- There is an NNPFC SEI message with `nnpfc_id` equal to the particular value of `nnpfa_target_id` in the current PU.

[00180] When a PU contains both an NNPFC SEI message with a particular value of `nnpfc_id` and an NNPFA SEI message with `nnpfa_target_id` equal to the particular value of `nnpfc_id`, the NNPFC SEI message shall precede the NNPFA SEI message in decoding order.

[00181] `nnpfa_cancel_flag` equal to 1 indicates that the persistence of the target NNPF established by any previous NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message is cancelled, i.e., the target NNPF is no longer used unless it is activated by another NNPFA SEI message with the same `nnpfa_target_id` as the current SEI message and `nnpfa_cancel_flag` equal to 0. `nnpfa_cancel_flag` equal to 0 indicates that the `nnpfa_persistence_flag` follows.

[00182] `nnpfa_persistence_flag` specifies the persistence of the target NNPF for the current layer.

[00183] `nnpfa_persistence_flag` equal to 0 specifies that the target NNPF may be used for post-processing filtering for the current picture only.

[00184] nnpfa_persistence_flag equal to 1 specifies that the target NNPF may be used for post-processing filtering for the current picture and all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.

5 – The bitstream ends.

- A picture in the current layer associated with a NNPF SEI message with the same nnpfa_target_id as the current SEI message and nnpfa_cancel_flag equal to 1 is output that follows the current picture in output order.

[00185] NOTE 2 – The target NNPF is not applied for this subsequent picture in the current
10 layer associated with a NNPF SEI message with the same nnpfa_target_id as the current SEI message and nnpfa_cancel_flag equal to 1.

[00186] Let the nnpfcTargetPictures be the set of pictures to which the last NNPF SEI message with nnpfc_id equal to nnpfa_target_id that precedes the current NNPF SEI message in decoding order pertains. Let nnpfaTargetPictures be the set of pictures for which the target
15 NNPF is activated by the current NNPF SEI message. It is a requirement of bitstream conformance that any picture included in nnpfaTargetPictures shall also be included in nnpfcTargetPictures.

4. Technical problems solved by disclosed technical solutions

[00187] An example design for the neural-network post-filter characteristics (NNPFC) SEI
20 message has the following problems:

[00188] First, neural-network post-filter purpose with resolution upsampling capability is specified. However, in video applications, often resolution downsampling is also needed, and like for resolution upsampling, using neural-network filters for resolution downsampling can also provide better performances than using conventional approaches. Therefore, there is a need
25 to be able to signal neural-network filters with resolution downsampling capability.

[00189] Second, neural-network post-filter purpose with bit depth upsampling capability is specified. However, in video applications, often bit depth downsampling is also needed, and like for bit depth upsampling, using neural-network filters for bit depth downsampling can also provide better performances than using conventional approaches. Therefore, there is a need to
30 be able to signal neural-network filters with bit depth downsampling capability.

[00190] Third, neural-network post-filter purpose with chroma upsampling capability is specified. However, in video applications, often chroma downsampling is also needed, and like for chroma upsampling, using neural-network filters for chroma downsampling can also

provide better performances than using conventional approaches. Therefore, there is a need to be able to signal neural-network filters with bit chroma downsampling capability.

[00191] Fourth, neural-network post-filter purpose with picture rate upsampling capability is specified. However, in video applications, often picture rate downsampling is also needed, and like for picture rate upsampling, using neural-network filters for picture rate downsampling can also provide better performances than using conventional approaches. Therefore, there is a need to be able to signal neural-network filters with picture rate downsampling capability.

5. A listing of solutions and embodiments

[00192] To solve the above-described problems, methods as summarized below are disclosed. The aspects should be considered as examples to explain the general concepts and should not be interpreted in a narrow way. Furthermore, these examples can be applied individually or combined in any manner.

1) To solve problem 1, one or more of the following new purposes are defined:

a. In one example, a new purpose for resolution downsampling is defined, with decreasing of either or both of the width and height of the cropped decoded output picture.

i. In one example, furthermore, it is required that a purpose shall not include both resolution upsampling and resolution downsampling.

ii. In one example, it is required that the downsampling ratio is restricted in a range of minR to maxR.

1. In one example, minR is equal to 1, 1/2, 1/4, 1/8, 1/16.

2. In one example, maxR is equal to 1.

b. In one example, a new purpose for resolution resampling is defined, with decreasing or increasing of either or both of the width and height of the cropped decoded output picture.

i. In one example, it is required that the resampling/scaling ratio is restricted in a range of minR to maxR.

1. In one example, minR is equal to 1, 1/2, 1/4, 1/8, 1/16.

2. In one example, maxR is equal to 1, 2, 4, 8, 16.

2) To solve problem 2, one or more of the following new purposes are defined:

a. In one example, a new purpose for bit depth downsampling is defined, with decreasing of either or both of the luma bit depth and chroma bit depth of the cropped decoded output picture.

i. In one example, furthermore, it is required that a purpose shall not include both bit depth upsampling and bit depth downsampling.

b. In one example, a new purpose for bit depth resampling is defined, with decreasing or increasing of either or both of the luma bit depth and chroma bit depth of the cropped decoded output picture.

3) To solve problem 3, one or more of the following new purposes are defined:

a. In one example, a new purpose for chroma format downsampling is defined, with changing the chroma format from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format.

i. In one example, furthermore, it is required that a purpose shall not include both chroma format upsampling and chroma format downsampling.

b. In one example, a new purpose for chroma format downsampling is defined, with changing the chroma format from the 4:4:4 chroma format to the 4:0:0, or from the 4:2:2 chroma format to the 4:0:0 chroma format, or from the 4:2:0 chroma format to the 4:0:0 chroma format.

c. In one example, a new purpose for chroma format resampling is defined, with changing the chroma format from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format, or from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format.

d. In one example, a new purpose for chroma format resampling is defined, with changing the chroma format from the 4:4:4 chroma format to the 4:2:2, 4:2:0, or 4:0:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 or 4:0:0 chroma format, or from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format, or from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format.

4) To solve problem 4, one or more of the following new purposes are defined:

a. In one example, a new purpose for picture rate downsampling is defined.

i. In one example, furthermore, it is required that a purpose shall not include both picture rate upsampling and picture rate downsampling.

b. In one example, a new purpose for picture rate resampling is defined.

6. Embodiments

[00193] Below are some example embodiments for the aspects summarized above in Section 5.

[00194] Most relevant parts that have been added or modified are in bold, and some of the deleted parts are in bold and italic fonts. There may be some other changes that are editorial in

5 nature and thus not indicated.

6.1 Embodiment 1

[00195] This embodiment is for items 1, 2, 3, 4 and all their subitems as summarized above in Section 5.

8.28.1 Neural-network post-filter characteristics SEI message syntax

nn_post_filter_characteristics(payloadSize) {	Descriptor
nnpfc_purpose	u(16)
nnpfc_id	ue(v)
nnpfc_mode_idc	ue(v)
if(nnpfc_mode_idc == 1) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_a	u(1)
nnpfc_tag_uri	st(v)
nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)
if(nnpfc_property_present_flag) {	
nnpfc_base_flag	u(1)
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if((nnpfc_purpose & 0x02) != 0 (nnpfc_purpose & 0x40) != 0)	
nnpfc_out_sub_c_flag	u(1)
if((nnpfc_purpose & 0x20) != 0)	
nnpfc_out_colour_format_idc	u(2)
if((nnpfc_purpose & 0x04) != 0 (nnpfc_purpose & 0x80) != 0)	

{	
nnpfc_pic_width_in_luma_samples	ue(v)
nnpfc_pic_height_in_luma_samples	ue(v)
}	
if((nnpfc_purpose & 0x08) != 0 (nnpfc_purpose & 0x100) != 0) {	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_output_flag[i]	u(1)
}	
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_inp_order_idc	ue(v)
nnpfc_auxiliary_inp_idc	ue(v)
nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	
nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
nnpfc_matrix_coeffs	u(8)
}	
nnpfc_out_format_idc	ue(v)
if(nnpfc_out_format_idc == 1) {	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)

}	
nnpfc_out_order_idc	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
nnpfc_luma_padding_val	ue(v)
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)
if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)
nnpfc_num_parameters_idc	u(6)
nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
}	
/* ISO/IEC 15938-17 bitstream */	

if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

8.28.2 Neural-network post-filter characteristics SEI message semantics

The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

Use of this SEI message requires the definition of the following variables:

- Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- 10 – Luma sample array CroppedYPic[idx] and chroma sample arrays CroppedCbPic[idx] and CroppedCrPic[idx], when present, of the input pictures with index idx in the range of 0 to numInputPics – 1, inclusive, that are used as input for the NNPF.
- Bit depth BitDepth_Y for the luma sample array of the input pictures.
- Bit depth BitDepth_C for the chroma sample arrays, if any, of the input pictures.
- 15 – A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause 7.3.
- When nnpfc_auxiliary_inp_idc is equal to 1, a filtering strength control value StrengthControlVal that shall be a real number in the range of 0 to 1, inclusive.

Input picture with index 0 corresponds to the picture for which the NNPF defined by this

- 20 NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in the range of 1 to numInputPics – 1, inclusive, precedes the input picture with index i – 1 in output order.

When nnpfc_purpose & 0x08 is not equal to 0 and the input picture with index 0 is associated with a frame packing arrangement SEI message with fp_arrangement_type equal to 5, all

input pictures are associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5 and the same value of `fp_current_frame_is_frame0_flag`. The variables `SubWidthC` and `SubHeightC` are derived from `ChromaFormatIdc` as specified by Table 2.

- 5 NOTE 1 – More than one NNPF SEI message can be present for the same picture. When more than one NNPF SEI message with different values of `nnpfc_id` is present or activated for the same picture, they can have the same or different values of `nnpfc_purpose` and `nnpfc_mode_idc`.

`nnpfc_purpose` indicates the purpose of the NNPF as specified in Table 20.

- 10 The value of `nnpfc_purpose` shall be in the range of 0 to **63 1023**, inclusive, in bitstreams conforming to this edition of this document. Values of **641024** to 65 535, inclusive, for `nnpfc_purpose` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_purpose` in the range of **641024**
15 to 65 535, inclusive.

Table 20 – Definition of `nnpfc_purpose`

Value	Interpretation
<code>nnpfc_purpose == 0</code>	May be used as determined by the application
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x01) == 0</code>	No general visual quality improvement
<code>(nnpfc_purpose & 0x01) != 0</code>	With general visual quality improvement
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x02) == 0</code>	No chroma upsampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format)
<code>(nnpfc_purpose & 0x02) != 0</code>	With chroma upsampling
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x04) == 0</code>	No resolution upsampling (increasing the width or height)

(nnpfc_purpose & 0x04) != 0	With resolution upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x08) == 0	No picture rate upsampling
(nnpfc_purpose & 0x08) != 0	With picture rate upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x10) == 0	No bit depth upsampling (increasing the luma bit depth or the chroma bit depth)
(nnpfc_purpose & 0x10) != 0	With bit depth upsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x20) == 0	No colourization (from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format)
(nnpfc_purpose & 0x20) != 0	With colourization
nnpfc_purpose > 0 && (nnpfc_purpose & 0x40) == 0	No chroma downsampling (from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format)
(nnpfc_purpose & 0x40) != 0	With chroma downsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x80) == 0	No resolution downsampling (decreasing the width or height)
(nnpfc_purpose & 0x80) != 0	With resolution downsampling
nnpfc_purpose > 0 &&	No picture rate downsampling

(nnpfc_purpose & 0x100) == 0	
(nnpfc_purpose & 0x100) != 0	With picture rate downsampling
nnpfc_purpose > 0 && (nnpfc_purpose & 0x200) == 0	No bit depth downsampling (decreasing the luma bit depth or the chroma bit depth)
(nnpfc_purpose & 0x200) != 0	With bit depth downsampling

NOTE 2– When a reserved value of nnpfc_purpose is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by nnpfc_purpose being equal to that value.

- 5 When ChromaFormatIdc is equal to 3, nnpfc_purpose & 0x02 shall be equal to 0.

When ChromaFormatIdc is equal to 1, nnpfc_purpose & 0x40 shall be equal to 0.

When ChromaFormatIdc or nnpfc_purpose & 0x02 is not equal to 0, nnpfc_purpose & 0x20 shall be equal to 0.

When ChromaFormatIdc or nnpfc_purpose & 0x40 is not equal to 0,

- 10 **nnpfc_purpose & 0x20 shall be equal to 0.**

When nnpfc_purpose & 0x02 is not equal to 0, nnpfc_purpose & 0x40 shall be equal to 0.

When nnpfc_purpose & 0x04 is not equal to 0, nnpfc_purpose & 0x80 shall be equal to 0.

- 15 **When nnpfc_purpose & 0x08 is not equal to 0, nnpfc_purpose & 0x100 shall be equal to 0.**

nnpfc_id contains an identifying number that may be used to identify an NNPF. The value of nnpfc_id shall be in the range of 0 to $2^{32} - 2$, inclusive. Values of nnpfc_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC.

- 20 Decoders conforming to this edition of this document encountering an NNPF SEI message with nnpfc_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore the SEI message.

When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a

particular nnpfc_id value within the current CLVS, the following applies:

- This SEI message specifies a base NNPF.
- This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS.

5 ...

nnpfc_out_colour_format_idc, when nnpfc_purpose & 0x20 is not equal to 0, specifies the colour format of the NNPF output and consequently the values of the variables outSubWidthC and outSubHeightC. nnpfc_out_colour_format_idc equal to 1 specifies that the colour format of the NNPF output is the 4:2:0 format and outSubWidthC and

10 outSubHeightC are both equal to 2. nnpfc_out_colour_format_idc equal to 2 specifies that the colour format of the NNPF output is the 4:2:2 format and outSubWidthC is equal to 2 and outSubHeightC is equal to 1. nnpfc_out_colour_format_idc equal to 3 specifies that the colour format of the NNPF output is the 4:2:4 format and outSubWidthC and outSubHeightC are both equal to 1. The value of nnpfc_out_colour_format_idc shall not be equal to 0.

15 When nnpfc_purpose & 0x02 and nnpfc_purpose & 0x20 **and nnpfc_purpose & 0x40** are both equal to 0, outSubWidthC and outSubHeightC are inferred to be equal to SubWidthC and SubHeightC, respectively.

nnpfc_pic_width_in_luma_samples and nnpfc_pic_height_in_luma_samples specify the width and height, respectively, of the luma sample array of the picture resulting from applying

20 the NNPF identified by nnpfc_id to a cropped decoded output picture. When

nnpfc_pic_width_in_luma_samples and nnpfc_pic_height_in_luma_samples are not present, they are inferred to be equal to CroppedWidth and CroppedHeight, respectively. The value of nnpfc_pic_width_in_luma_samples shall be in the range of $\text{CroppedWidth} \gg 4$ to

25 $\text{CroppedWidth} * 16 - 1$, inclusive. The value of nnpfc_pic_height_in_luma_samples shall be in the range of $\text{CroppedHeight} \gg 4$ to $\text{CroppedHeight} * 16 - 1$, inclusive.

When nnpfc_purpose & 0x04 is not equal to 0, at least one of the following conditions shall be true:

- **nnpfc_pic_width_in_luma_samples is greater than CroppedWidth and nnpfc_pic_height_in_luma_samples is not less than CroppedHeight.**

30 – **nnpfc_pic_height_in_luma_samples is greater than CroppedHeight and nnpfc_pic_width_in_luma_samples is not less than CroppedWidth.**

When nnpfc_purpose & 0x80 is not equal to 0, at least one of the following conditions shall be true:

- **nnpfc_pic_width_in_luma_samples is less than CroppedWidth and nnpfc_pic_height_in_luma_samples is not greater than CroppedHeight.**
- **nnpfc_pic_height_in_luma_samples is less than CroppedHeight and nnpfc_pic_width_in_luma_samples is not greater than CroppedWidth.**

5 nnpfc_num_input_pics_minus1 plus 1 specifies the number of decoded output pictures used as input for the NNPF. The value of nnpfc_num_input_pics_minus1 shall be in the range of 0 to 63, inclusive. When nnpfc_purpose & 0x08 is not equal to 0, the value of nnpfc_num_input_pics_minus1 shall be greater than 0.

...

10 nnpfc_out_format_idc equal to 0 indicates that the sample values output by the NNPF are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth bitDepth for subsequent post-processing or displaying.

nnpfc_out_format_idc equal to 1 indicates that the luma sample values output by the NNPF
15 are unsigned integer numbers in the range of 0 to

$(1 \ll (\text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8)) - 1$, inclusive, and the chroma sample values output by the NNPF are unsigned integer numbers in the range of 0 to $(1 \ll (\text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8)) - 1$, inclusive.

Values of nnpfc_out_format_idc greater than 1 are reserved for future specification by ITU-T
20 | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document.

Decoders conforming to this edition of this document shall ignore NNPF SEI messages that contain reserved values of nnpfc_out_format_idc.

nnpfc_out_tensor_luma_bitdepth_minus8 plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of nnpfc_out_tensor_luma_bitdepth_minus8 shall be in
25 the range of 0 to 24, inclusive.

nnpfc_out_tensor_chroma_bitdepth_minus8 plus 8 specifies the bit depth of chroma sample values in the output integer tensor. The value of nnpfc_out_tensor_chroma_bitdepth_minus8 shall be in the range of 0 to 24, inclusive.

When nnpfc_purpose & 0x10 is not equal to 0, the value of nnpfc_out_format_idc shall be
30 equal to 1 and at least one of the following conditions shall be true:

- **nnpfc_out_tensor_luma_bitdepth_minus8 + 8 is greater than BitDepth_Y and nnpfc_out_tensor_chroma_bitdepth_minus8 + 8 is not less than BitDepth_C.**
- **nnpfc_out_tensor_chroma_bitdepth_minus8 + 8 is greater than BitDepth_C and**

nnpfc_out_tensor_luma_bitdepth_minus8 + 8 is not less than BitDepth_Y.

When **nnpfc_purpose & 0x200** is not equal to 0, the value of **nnpfc_out_format_idc** shall be equal to 1 and at least one of the following conditions shall be true:

- **nnpfc_out_tensor_luma_bitdepth_minus8 + 8 is less than BitDepth_Y and**
- 5 **nnpfc_out_tensor_chroma_bitdepth_minus8 + 8 is not greater than BitDepth_C.**
- **nnpfc_out_tensor_chroma_bitdepth_minus8 + 8 is less than BitDepth_C and**
- nnpfc_out_tensor_luma_bitdepth_minus1 + 1 is not greater than BitDepth_Y.**

nnpfc_out_order_idc indicates the output order of samples resulting from the NNPF.

The value of **nnpfc_out_order_idc** shall be in the range of 0 to 3, inclusive, in bitstreams

- 10 conforming to this edition of this document. Values of 4 to 255, inclusive, for **nnpfc_out_order_idc** are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with **nnpfc_out_order_idc** in the range of 4 to 255, inclusive. Values of **nnpfc_out_order_idc** greater than 255 shall not be present in
- 15 bitstreams conforming to this edition of this document and are not reserved for future use.

...

6.2 Embodiment 2

[00196] This embodiment is for items 1, 2, 3, 4, and all their subitems as summarized above in Section 5.

20 8.28.1 Neural-network post-filter characteristics SEI message syntax

nn_post_filter_characteristics(payloadSize) {	Descriptor
nnpfc_purpose	u(16)
nnpfc_id	ue(v)
nnpfc_mode_idc	ue(v)
if(nnpfc_mode_idc == 1) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_a	u(1)
nnpfc_tag_uri	st(v)
nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)

if(nnpfc_property_present_flag) {	
nnpfc_base_flag	u(1)
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if((nnpfc_purpose & 0x02) != 0)	
nnpfc_out_sub_c_flag	u(1)
if((nnpfc_purpose & 0x20) != 0)	
nnpfc_out_colour_format_idc	u(2)
if((nnpfc_purpose & 0x04) != 0) {	
nnpfc_pic_width_in_luma_samples	ue(v)
nnpfc_pic_height_in_luma_samples	ue(v)
}	
if((nnpfc_purpose & 0x08) != 0) {	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_output_flag[i]	u(1)
}	
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_inp_order_idc	ue(v)
nnpfc_auxiliary_inp_idc	ue(v)
nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	

nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
nnpfc_matrix_coeffs	u(8)
}	
nnpfc_out_format_idc	ue(v)
if(nnpfc_out_format_idc == 1) {	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_out_order_idc	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
nnpfc_luma_padding_val	ue(v)
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)

if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)
nnpfc_num_parameters_idc	u(6)
nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
}	
/* ISO/IEC 15938-17 bitstream */	
if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

8.28.2 Neural-network post-filter characteristics SEI message semantics

The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

Use of this SEI message requires the definition of the following variables:

- Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- Luma sample array CroppedYPic[idx] and chroma sample arrays CroppedCbPic[idx] and CroppedCrPic[idx], when present, of the input pictures with index idx in the range of 0 to numInputPics – 1, inclusive, that are used as input for the NNPF.
- Bit depth BitDepth_y for the luma sample array of the input pictures.
- Bit depth BitDepth_c for the chroma sample arrays, if any, of the input pictures.
- A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause 7.3.

- When `nnpfc_auxiliary_inp_idc` is equal to 1, a filtering strength control value `StrengthControlVal` that shall be a real number in the range of 0 to 1, inclusive.

Input picture with index 0 corresponds to the picture for which the NNPF defined by this NNPF SEI message is activated by an NNPF SEI message. Input picture with index *i* in the range of 1 to `numInputPics` – 1, inclusive, precedes the input picture with index *i* – 1 in output order.

When `nnpfc_purpose & 0x08` is not equal to 0 and the input picture with index 0 is associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5, all input pictures are associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5 and the same value of `fp_current_frame_is_frame0_flag`. The variables `SubWidthC` and `SubHeightC` are derived from `ChromaFormatIdc` as specified by Table 2.

NOTE 1 – More than one NNPF SEI message can be present for the same picture. When more than one NNPF SEI message with different values of `nnpfc_id` is present or activated for the same picture, they can have the same or different values of `nnpfc_purpose` and `nnpfc_mode_idc`.

`nnpfc_purpose` indicates the purpose of the NNPF as specified in Table 20.

The value of `nnpfc_purpose` shall be in the range of 0 to 63, inclusive, in bitstreams conforming to this edition of this document. Values of 64 to 65 535, inclusive, for `nnpfc_purpose` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_purpose` in the range of 64 to 65 535, inclusive.

Table 20 – Definition of `nnpfc_purpose`

Value	Interpretation
<code>nnpfc_purpose == 0</code>	May be used as determined by the application
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x01) == 0</code>	No general visual quality improvement
<code>(nnpfc_purpose & 0x01) != 0</code>	With general visual quality improvement

$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x02) == 0$	No chroma <i>up</i> resampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format, or from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format)
$(\text{nnpfc_purpose} \ \& \ 0x02) != 0$	With chroma <i>up</i> resampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x04) == 0$	No resolution <i>up</i> resampling (increasing or deceasing the width or height)
$(\text{nnpfc_purpose} \ \& \ 0x04) != 0$	With resolution <i>up</i> resampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x08) == 0$	No picture rate <i>up</i> resampling
$(\text{nnpfc_purpose} \ \& \ 0x08) != 0$	With picture rate <i>up</i> resampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x10) == 0$	No bit depth <i>up</i> resampling (increasing or deceasing the luma bit depth or the chroma bit depth)
$(\text{nnpfc_purpose} \ \& \ 0x10) != 0$	With bit depth <i>up</i> resampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x20) == 0$	No colourization (from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format)
$(\text{nnpfc_purpose} \ \& \ 0x20) != 0$	With colourization

NOTE 2 – When a reserved value of `nnpfc_purpose` is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by `nnpfc_purpose` being equal to that value.

When ChromaFormatIdc is equal to 3, nnpfc_purpose & 0x02 shall be equal to 0.

When ChromaFormatIdc or nnpfc_purpose & 0x02 is not equal to 0, nnpfc_purpose & 0x20 shall be equal to 0.

nnpfc_id contains an identifying number that may be used to identify an NNPF. The value of nnpfc_id shall be in the range of 0 to $2^{32} - 2$, inclusive. Values of nnpfc_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPF SEI message with nnpfc_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore the SEI message.

10 When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a particular nnpfc_id value within the current CLVS, the following applies:

- This SEI message specifies a base NNPF.
- This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS.

15 ...

nnpfc_out_colour_format_idc, when nnpfc_purpose & 0x20 is not equal to 0, specifies the colour format of the NNPF output and consequently the values of the variables outSubWidthC and outSubHeightC. nnpfc_out_colour_format_idc equal to 1 specifies that the colour format of the NNPF output is the 4:2:0 format and outSubWidthC and

20 outSubHeightC are both equal to 2. nnpfc_out_colour_format_idc equal to 2 specifies that the colour format of the NNPF output is the 4:2:2 format and outSubWidthC is equal to 2 and outSubHeightC is equal to 1. nnpfc_out_colour_format_idc equal to 3 specifies that the colour format of the NNPF output is the 4:2:4 format and outSubWidthC and outSubHeightC are both equal to 1. The value of nnpfc_out_colour_format_idc shall not be equal to 0.

25 When nnpfc_purpose & 0x02 and nnpfc_purpose & 0x20 are both equal to 0, outSubWidthC and outSubHeightC are inferred to be equal to SubWidthC and SubHeightC, respectively. nnpfc_pic_width_in_luma_samples and nnpfc_pic_height_in_luma_samples specify the width and height, respectively, of the luma sample array of the picture resulting from applying the NNPF identified by nnpfc_id to a cropped decoded output picture. When

30 nnpfc_pic_width_in_luma_samples and nnpfc_pic_height_in_luma_samples are not present, they are inferred to be equal to CroppedWidth and CroppedHeight, respectively. The value of nnpfc_pic_width_in_luma_samples shall be in the range of $\text{CroppedWidth} \gg 4$ to $\text{CroppedWidth} * 16 - 1$, inclusive. The value of nnpfc_pic_height_in_luma_samples shall be

in the range of $\text{CroppedHeight} \gg 4$ to $\text{CroppedHeight} * 16 - 1$, inclusive.

When $\text{nnpfc_purpose} \& 0x04$ is not equal to 0, at least one of the following conditions shall be true:

- **($\text{nnpfc_pic_width_in_luma_samples}$ is greater than CroppedWidth and
5 $\text{nnpfc_pic_height_in_luma_samples}$ is not less than CroppedHeight), or
($\text{nnpfc_pic_height_in_luma_samples}$ is greater than CroppedHeight and
 $\text{nnpfc_pic_width_in_luma_samples}$ is not less than CroppedWidth).**
- **($\text{nnpfc_pic_width_in_luma_samples}$ is less than CroppedWidth and
 $\text{nnpfc_pic_height_in_luma_samples}$ is not greater than CroppedHeight), or
10 **($\text{nnpfc_pic_height_in_luma_samples}$ is less than CroppedHeight and
 $\text{nnpfc_pic_width_in_luma_samples}$ is not greater than CroppedWidth).****

$\text{nnpfc_num_input_pics_minus1}$ plus 1 specifies the number of decoded output pictures used as input for the NNPF. The value of $\text{nnpfc_num_input_pics_minus1}$ shall be in the range of 0 to 63, inclusive. When $\text{nnpfc_purpose} \& 0x08$ is not equal to 0, the value of

- 15 $\text{nnpfc_num_input_pics_minus1}$ shall be greater than 0.

...

- $\text{nnpfc_out_format_idc}$ equal to 0 indicates that the sample values output by the NNPF are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth bitDepth for
20 subsequent post-processing or displaying.

$\text{nnpfc_out_format_idc}$ equal to 1 indicates that the luma sample values output by the NNPF are unsigned integer numbers in the range of 0 to

- $(1 \ll (\text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8)) - 1$, inclusive, and the chroma sample values output by the NNPF are unsigned integer numbers in the range of 0 to
25 $(1 \ll (\text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8)) - 1$, inclusive.

Values of $\text{nnpfc_out_format_idc}$ greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document.

Decoders conforming to this edition of this document shall ignore NNPF SEI messages that contain reserved values of $\text{nnpfc_out_format_idc}$.

- 30 $\text{nnpfc_out_tensor_luma_bitdepth_minus8}$ plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of $\text{nnpfc_out_tensor_luma_bitdepth_minus8}$ shall be in the range of 0 to 24, inclusive.

$\text{nnpfc_out_tensor_chroma_bitdepth_minus8}$ plus 8 specifies the bit depth of chroma sample

values in the output integer tensor. The value of `nnpfc_out_tensor_chroma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

When `nnpfc_purpose & 0x10` is not equal to 0, the value of `nnpfc_out_format_idc` shall be equal to 1 and at least one of the following conditions shall be true:

- 5 – (`nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is greater than `BitDepthY` **and**
 `nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is not less than `BitDepthC`), or
 (`nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is greater than `BitDepthC` **and**
 `nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is not less than `BitDepthY`)
- (`nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is less than `BitDepthY` **and**
 `nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is not greater than `BitDepthC`) or
 (`nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is less than `BitDepthC` **and**
 `nnpfc_out_tensor_luma_bitdepth_minus1 + 1` is not greater than `BitDepthY`).

`nnpfc_out_order_idc` indicates the output order of samples resulting from the NNPF.

The value of `nnpfc_out_order_idc` shall be in the range of 0 to 3, inclusive, in bitstreams

- 15 conforming to this edition of this document. Values of 4 to 255, inclusive, for
 `nnpfc_out_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present
 in bitstreams conforming to this edition of this document. Decoders conforming to this edition
 of this document shall ignore NNPF SEI messages with `nnpfc_out_order_idc` in the range of
 4 to 255, inclusive. Values of `nnpfc_out_order_idc` greater than 255 shall not be present in
 bitstreams conforming to this edition of this document and are not reserved for future use.

...

6.3 Embodiment 3

[00197] This embodiment is for items 1, 2, 3, 4, and all their subitems as summarized above in Section 5.

- 25 8.28.1 Neural-network post-filter characteristics SEI message syntax

<code>nn_post_filter_characteristics(payloadSize) {</code>	Descriptor
<code>nnpfc_purpose</code>	<code>u(16)</code>
<code>nnpfc_id</code>	<code>ue(v)</code>
<code>nnpfc_mode_idc</code>	<code>ue(v)</code>
<code>if(nnpfc_mode_idc == 1) {</code>	
<code>while(!byte_aligned())</code>	
<code>nnpfc_reserved_zero_bit_a</code>	<code>u(1)</code>

nnpfc_tag_uri	st(v)
nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)
if(nnpfc_property_present_flag) {	
nnpfc_base_flag	u(1)
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if((nnpfc_purpose & 0x02) != 0)	
nnpfc_out_sub_c_flag	u(1)
if((nnpfc_purpose & 0x20) != 0)	
nnpfc_out_colour_format_idc	u(2)
if((nnpfc_purpose & 0x04) != 0) {	
nnpfc_hor_ratio_num_minus1	u(8)
nnpfc_hor_ratio_den_minus1	u(8)
nnpfc_ver_ratio_num_minus1	u(8)
nnpfc_ver_ratio_den_minus1	u(8)
<i>nnpfc_pic_width_in_luma_samples</i>	<i>ue(v)</i>
<i>nnpfc_pic_height_in_luma_samples</i>	<i>ue(v)</i>
}	
if((nnpfc_purpose & 0x08) != 0) {	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_output_flag[i]	u(1)
}	
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_inp_order_idc	ue(v)

nnpfc_auxiliary_inp_idc	ue(v)
nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	
nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
nnpfc_matrix_coeffs	u(8)
}	
nnpfc_out_format_idc	ue(v)
if(nnpfc_out_format_idc == 1) {	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_out_order_idc	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
nnpfc_luma_padding_val	ue(v)
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)
if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)

nnpfc_num_parameters_idc	u(6)
nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
}	
/* ISO/IEC 15938-17 bitstream */	
if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

8.28.2 Neural-network post-filter characteristics SEI message semantics

The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network

5 post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

Use of this SEI message requires the definition of the following variables:

- Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- 10 – Luma sample array CroppedYPic[idx] and chroma sample arrays CroppedCbPic[idx] and CroppedCrPic[idx], when present, of the input pictures with index idx in the range of 0 to numInputPics – 1, inclusive, that are used as input for the NNPF.
- Bit depth BitDepth_Y for the luma sample array of the input pictures.
- Bit depth BitDepth_C for the chroma sample arrays, if any, of the input pictures.
- 15 – A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause 7.3.
- When nnpfc_auxiliary_inp_idc is equal to 1, a filtering strength control value StrengthControlVal that shall be a real number in the range of 0 to 1, inclusive.

Input picture with index 0 corresponds to the picture for which the NNPF defined by this

20 NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in

the range of 1 to numInputPics – 1, inclusive, precedes the input picture with index $i - 1$ in output order.

When nnpfc_purpose & 0x08 is not equal to 0 and the input picture with index 0 is associated with a frame packing arrangement SEI message with fp_arrangement_type equal to 5, all

5 input pictures are associated with a frame packing arrangement SEI message with fp_arrangement_type equal to 5 and the same value of fp_current_frame_is_frame0_flag.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

NOTE 1 – More than one NNPF SEI message can be present for the same picture. When
10 more than one NNPF SEI message with different values of nnpfc_id is present or activated for the same picture, they can have the same or different values of nnpfc_purpose and nnpfc_mode_idc.

nnpfc_purpose indicates the purpose of the NNPF as specified in Table 20.

The value of nnpfc_purpose shall be in the range of 0 to 63, inclusive, in bitstreams
15 conforming to this edition of this document. Values of 64 to 65 535, inclusive, for nnpfc_purpose are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with nnpfc_purpose in the range of 64 to 65 535, inclusive.

20 Table 20 – Definition of nnpfc_purpose

Value	Interpretation
nnpfc_purpose == 0	May be used as determined by the application
nnpfc_purpose > 0 && (nnpfc_purpose & 0x01) == 0	No general visual quality improvement
(nnpfc_purpose & 0x01) != 0	With general visual quality improvement
nnpfc_purpose > 0 && (nnpfc_purpose & 0x02) == 0	No chroma up resampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format, or from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format)
(nnpfc_purpose &	With chroma up resampling

$0x02) \neq 0$	
$nnpfc_purpose > 0 \ \&\& \ (nnpfc_purpose \ \& \ 0x04) == 0$	No resolution upresampling (increasing or decreasing the width or height)
$(nnpfc_purpose \ \& \ 0x04) \neq 0$	With resolution upresampling
$nnpfc_purpose > 0 \ \&\& \ (nnpfc_purpose \ \& \ 0x08) == 0$	No picture rate upresampling
$(nnpfc_purpose \ \& \ 0x08) \neq 0$	With picture rate upresampling
$nnpfc_purpose > 0 \ \&\& \ (nnpfc_purpose \ \& \ 0x10) == 0$	No bit depth upresampling (increasing or decreasing the luma bit depth or the chroma bit depth)
$(nnpfc_purpose \ \& \ 0x10) \neq 0$	With bit depth upresampling
$nnpfc_purpose > 0 \ \&\& \ (nnpfc_purpose \ \& \ 0x20) == 0$	No colourization (from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format)
$(nnpfc_purpose \ \& \ 0x20) \neq 0$	With colourization

NOTE 2 – When a reserved value of `nnpfc_purpose` is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by `nnpfc_purpose` being equal to that value.

5 ***When ChromaFormatIdc is equal to 3, nnpfc_purpose & 0x02 shall be equal to 0.***

When `ChromaFormatIdc` or `nnpfc_purpose & 0x02` is not equal to 0, `nnpfc_purpose & 0x20` shall be equal to 0.

`nnpfc_id` contains an identifying number that may be used to identify an NNPF. The value of `nnpfc_id` shall be in the range of 0 to $2^{32} - 2$, inclusive. Values of `nnpfc_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC.

10 Decoders conforming to this edition of this document encountering an NNPF SEI message with `nnpfc_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive,

shall ignore the SEI message.

When an NNPFC SEI message is the first NNPFC SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, the following applies:

- This SEI message specifies a base NNPFC.
- 5 – This SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS.

...

`nnpfc_out_colour_format_idc`, when `nnpfc_purpose & 0x20` is not equal to 0, specifies the colour format of the NNPFC output and consequently the values of the variables

- 10 `outSubWidthC` and `outSubHeightC`. `nnpfc_out_colour_format_idc` equal to 1 specifies that the colour format of the NNPFC output is the 4:2:0 format and `outSubWidthC` and `outSubHeightC` are both equal to 2. `nnpfc_out_colour_format_idc` equal to 2 specifies that the colour format of the NNPFC output is the 4:2:2 format and `outSubWidthC` is equal to 2 and `outSubHeightC` is equal to 1. `nnpfc_out_colour_format_idc` equal to 3 specifies that the
- 15 colour format of the NNPFC output is the 4:2:4 format and `outSubWidthC` and `outSubHeightC` are both equal to 1. The value of `nnpfc_out_colour_format_idc` shall not be equal to 0.

When `nnpfc_purpose & 0x02` and `nnpfc_purpose & 0x20` are both equal to 0, `outSubWidthC` and `outSubHeightC` are inferred to be equal to `SubWidthC` and `SubHeightC`, respectively.

- 20 ***`nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` specify the width and height, respectively, of the luma sample array of the picture resulting from applying the NNPFC identified by `nnpfc_id` to a cropped decoded output picture. When `nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` are not present, they are inferred to be equal to `CroppedWidth` and `CroppedHeight`, respectively. The value of `nnpfc_pic_width_in_luma_samples` shall be in the range of `CroppedWidth` to***
- 25 ***`CroppedWidth * 16`, inclusive. The value of `nnpfc_pic_height_in_luma_samples` shall be in the range of `CroppedHeight >> 4` to `CroppedHeight * 16`, inclusive.***

`nnpfc_hor_ratio_num_minus1` and `nnpfc_hor_ratio_den_minus1` specify the horizontal luma scaling ratio `nnpfcHorScalingRatio` as follows:

- `nnpfcHorScalingRatio` =**
- 30 **$(\text{nnpfc_hor_ratio_num_minus1} + 1) \div (\text{nnpfc_hor_ratio_den_minus1} + 1) \quad (X)$**

The value of `nnpfcHorScalingRatio` shall be in the range of 1/16 to 16, inclusive. When `nnpfc_hor_ratio_num_minus1` and `nnpfc_hor_ratio_den_minus1` are not present, the value of `nnpfcHorScalingRatio` is inferred to be equal to 1.

The value of the variable `nnpfcOutputPicWidth`, representing the width of the luma

sample array of the picture(s) resulting from applying the NNPF identified by `nnpfc_id` to a cropped decoded output picture, is derived as follows:

$$\text{nnpfcOutputPicWidth} = \text{CroppedWidth} * \text{nnpfcHorScalingRatio} \quad (\text{X})$$

`nnpfc_ver_ratio_num_minus1` and `nnpfc_ver_ratio_den_minus1` specify the vertical

5 luma scaling ratio `nnpfcVerScalingRatio` is derived as follows:

$$\text{nnpfcVerScalingRatio} =$$

$$(\text{nnpfc_ver_ratio_num_minus1} + 1) \div (\text{nnpfc_ver_ratio_den_minus1} + 1) \quad (\text{X})$$

The value of `nnpfcVerScalingRatio` shall be in the range of 1/16 to 16, inclusive. When

`nnpfc_ver_ratio_num_minus1` and `nnpfc_ver_ratio_den_minus1` are not present, the

10 value of `nnpfcVerScalingRatio` is inferred to be equal to 1.

The value of the variable `nnpfcOutputPicHeight`, representing the height of the luma sample array of the picture(s) resulting from applying the NNPF identified by `nnpfc_id` to a cropped decoded output picture, is derived as follows:

$$\text{nnpfcOutputPicHeight} = \text{CroppedHeight} * \text{nnpfcHorScalingRatio} \quad (\text{X})$$

15 When `nnpfc_purpose & 0x04` is not equal to 0, at least one of the following conditions shall be true:

– `nnpfcOutputPicWidth` is greater than `CroppedWidth` and `nnpfcOutputPicHeight` is not less than `CroppedHeight`, or `nnpfcOutputPicHeight` is greater than `CroppedHeight` and `nnpfcOutputPicWidth` is not less than `CroppedWidth`.

20 – `nnpfcOutputPicWidth` is less than `CroppedWidth` and `nnpfcOutputPicHeight` is not greater than `CroppedHeight`, or `nnpfcOutputPicHeight` is less than `CroppedHeight` and `nnpfcOutputPicWidth` is not greater than `CroppedWidth`.

`nnpfc_num_input_pics_minus1` plus 1 specifies the number of decoded output pictures used as input for the NNPF. The value of `nnpfc_num_input_pics_minus1` shall be in the range of 0

25 to 63, inclusive. When `nnpfc_purpose & 0x08` is not equal to 0, the value of `nnpfc_num_input_pics_minus1` shall be greater than 0.

...

`nnpfc_out_format_idc` equal to 0 indicates that the sample values output by the NNPF are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth `bitDepth` for subsequent post-processing or displaying.

30 `nnpfc_out_format_idc` equal to 1 indicates that the luma sample values output by the NNPF are unsigned integer numbers in the range of 0 to

$(1 \ll (\text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8)) - 1$, inclusive, and the chroma

sample values output by the NNPF are unsigned integer numbers in the range of 0 to $(1 \ll (\text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8)) - 1$, inclusive.

Values of `nnpfc_out_format_idc` greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document.

- 5 Decoders conforming to this edition of this document shall ignore NNPF SEI messages that contain reserved values of `nnpfc_out_format_idc`.

`nnpfc_out_tensor_luma_bitdepth_minus8` plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of `nnpfc_out_tensor_luma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

- 10 `nnpfc_out_tensor_chroma_bitdepth_minus8` plus 8 specifies the bit depth of chroma sample values in the output integer tensor. The value of `nnpfc_out_tensor_chroma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

When `nnpfc_purpose & 0x10` is not equal to 0, the value of `nnpfc_out_format_idc` shall be equal to 1 and at least one of the following conditions shall be true:

- 15 – (`nnpfc_out_tensor_luma_bitdepth_minus8` + 8 is greater than `BitDepthY` **and** `nnpfc_out_tensor_chroma_bitdepth_minus8` + 8 is not less than `BitDepthC`), or
 (`nnpfc_out_tensor_chroma_bitdepth_minus8` + 8 is greater than `BitDepthC` **and** `nnpfc_out_tensor_luma_bitdepth_minus8` + 8 is not less than `BitDepthY`),
 – (`nnpfc_out_tensor_luma_bitdepth_minus8` + 8 is less than `BitDepthY` **and**
 20 `nnpfc_out_tensor_chroma_bitdepth_minus8` + 8 is not greater than `BitDepthC`), or
 (`nnpfc_out_tensor_chroma_bitdepth_minus8` + 8 is less than `BitDepthC` **and** `nnpfc_out_tensor_luma_bitdepth_minus1` + 1 is not greater than `BitDepthY`).

`nnpfc_out_order_idc` indicates the output order of samples resulting from the NNPF.

The value of `nnpfc_out_order_idc` shall be in the range of 0 to 3, inclusive, in bitstreams

- 25 conforming to this edition of this document. Values of 4 to 255, inclusive, for `nnpfc_out_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_out_order_idc` in the range of 4 to 255, inclusive. Values of `nnpfc_out_order_idc` greater than 255 shall not be present in
 30 bitstreams conforming to this edition of this document and are not reserved for future use.

When `nnpfc_purpose & 0x02` is not equal to 0, `nnpfc_out_order_idc` shall not be equal to 3.

Table 22 contains an informative description of `nnpfc_out_order_idc` values.

Table 22 – Description of `nnpfc_out_order_idc` values

nnpfc_out_order_idc	Description
0	Only the luma matrix is present in the output tensor, thus the number of channels is 1.
1	Only the chroma matrices are present in the output tensor, thus the number of channels is 2.
2	The luma and chroma matrices are present in the output tensor, thus the number of channels is 3.
3	Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This nnpfc_out_order_idc can only be used when the output chroma format is 4:2:0.
4..255	Reserved

The process StoreOutputTensors(), for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensor outputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

for( i = 0; i < numOutputPics; i++ ) {
  if( nnpfc_out_order_idc == 0 )
    for( yP = 0; yP < outPatchHeight; yP++ )
      for( xP = 0; xP < outPatchWidth; xP++ ) {
        yY = cTop * outPatchHeight / inpPatchHeight + yP
        xY = cLeft * outPatchWidth / inpPatchWidth + xP
        if ( yY < nnpfc_pic_height_in_luma_samples && xY <
nnpfc_pic_width_in_luma_samples )
15      if ( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth )
        if( !nnpfc_component_last_flag )
          FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
        else
          FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
20      }
    else if( nnpfc_out_order_idc == 1 )

```

(85)

```

for( yP = 0; yP < outPatchCHeight; yP++)
  for( xP = 0; xP < outPatchCWidth; xP++ ) {
    xSrc = cLeft * horCScaling + xP
    ySrc = cTop * verCScaling + yP
5    if ( ySrc < nnpfc_pic_height_in_luma_samples / outSubHeightC &&
      xSrc < nnpfc_pic_width_in_luma_samples / outSubWidthC )
      if ( ySrc < nnpfcOutputPicHeight / outSubHeightC &&
        xSrc < nnpfcOutputPicWidth / outSubWidthC )
        if( !nnpfc_component_last_flag ) {
10          FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
          FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
        } else {
          FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
          FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
15        }
      }
    }
  }
else if( nnpfc_out_order_idc == 2 )
  for( yP = 0; yP < outPatchHeight; yP++)
    for( xP = 0; xP < outPatchWidth; xP++ ) {
20      yY = cTop * outPatchHeight / inpPatchHeight + yP
      xY = cLeft * outPatchWidth / inpPatchWidth + xP
      yC = yY / outSubHeightC
      xC = xY / outSubWidthC
      yPc = ( yP / outSubHeightC ) * outSubHeightC
25      xPc = ( xP / outSubWidthC ) * outSubWidthC
      if ( yY < nnpfc_pic_height_in_luma_samples && xY <
nnpfc_pic_width_in_luma_samples)
        if ( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth)
          if( !nnpfc_component_last_flag ) {
30            FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
            FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 1 ][ yPc ][ xPc ]
            FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 2 ][ yPc ][ xPc ]
          } else {
            FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]

```

```

        FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 1 ]
        FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 2 ]
    }
}

5   else if( nnpfc_out_order_idc == 3 )
    for( yP = 0; yP < outPatchHeight; yP++ )
        for( xP = 0; xP < outPatchWidth; xP++ ) {
            ySrc = cTop / 2 * outPatchHeight / inpPatchHeight + yP
            xSrc = cLeft / 2 * outPatchWidth / inpPatchWidth + xP
10      if ( ySrc < nnpfc_pic_height_in_luma_samples / 2 &&
            xSrc < nnpfc_pic_width_in_luma_samples / 2 )
            if ( ySrc < nnpfcOutputPicHeight / 2 &&
            xSrc < nnpfcOutputPicWidth / 2 )
                if( !nnpfc_component_last_flag ) {
15          FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
          FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
          FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
20      outputTensor[ 0 ][ i ][ 2 ][ yP ][ xP ]
          FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ 3 ][ yP ][ xP ]
          FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 4 ][ yP ][ xP ]
          FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 5 ][ yP ][ xP ]
25      } else {
          FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
          FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
30          FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 2 ]
          FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =
outputTensor[ 0 ][ i ][ yP ][ xP ][ 3 ]
          FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 4 ]

```

```

        FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 5 ]
    }
}
}

```

- 5 nnpfc_overlap indicates the overlapping horizontal and vertical sample counts of adjacent input tensors of the NNPF. The value of nnpfc_overlap shall be in the range of 0 to 16 383, inclusive.
- nnpfc_constant_patch_size_flag equal to 1 indicates that the NNPF accepts exactly the patch size indicated by nnpfc_patch_width_minus1 and nnpfc_patch_height_minus1 as input.
- 10 nnpfc_constant_patch_size_flag equal to 0 indicates that the NNPF accepts as input any patch size with width inpPatchWidth and height inpPatchHeight such that the width of an extended patch (i.e., a patch plus the overlapping area), which is equal to $\text{inpPatchWidth} + 2 * \text{nnpfc_overlap}$, is a positive integer multiple of $\text{nnpfc_extended_patch_width_cd_delta_minus1} + 1 + 2 * \text{nnpfc_overlap}$, and the height of the
- 15 extended patch, which is equal to $\text{inpPatchHeight} + 2 * \text{nnpfc_overlap}$, is a positive integer multiple of $\text{nnpfc_extended_patch_height_cd_delta_minus1} + 1 + 2 * \text{nnpfc_overlap}$. nnpfc_patch_width_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to 1, indicates the horizontal sample counts of the patch size required for the input to the NNPF. The value of nnpfc_patch_width_minus1 shall be in the range of 0 to $\text{Min}(32\,766, \text{CroppedWidth} - 1)$, inclusive.
- 20 nnpfc_patch_height_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to 1, indicates the vertical sample counts of the patch size required for the input to the NNPF. The value of nnpfc_patch_height_minus1 shall be in the range of 0 to $\text{Min}(32\,766, \text{CroppedHeight} - 1)$, inclusive.
- 25 nnpfc_extended_patch_width_cd_delta_minus1 plus 1 plus $2 * \text{nnpfc_overlap}$, when nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the width of an extended patch required for the input to the NNPF. The value of nnpfc_extended_patch_width_cd_delta_minus1 shall be in the range of 0 to $\text{Min}(32\,766, \text{CroppedWidth} - 1)$, inclusive.
- 30 nnpfc_extended_patch_height_cd_delta_minus1 plus 1 plus $2 * \text{nnpfc_overlap}$, when nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the height of an extended patch required for the input to the NNPF. The value of nnpfc_extended_patch_height_cd_delta_minus1 shall be in the range of 0 to $\text{Min}(32\,766, \text{CroppedHeight} - 1)$, inclusive.

Let the variables `inpPatchWidth` and `inpPatchHeight` be the patch size width and the patch size height, respectively.

If `nnpfc_constant_patch_size_flag` is equal to 0, the following applies:

- The values of `inpPatchWidth` and `inpPatchHeight` are either provided by external means not specified in this document or set by the post-processor itself.
- The value of `inpPatchWidth + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchWidth` shall be less than or equal to `CroppedWidth`. The value of `inpPatchHeight + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchHeight` shall be less than or equal to `CroppedHeight`.

Otherwise (`nnpfc_constant_patch_size_flag` is equal to 1), the value of `inpPatchWidth` is set equal to `nnpfc_patch_width_minus1 + 1` and the value of `inpPatchHeight` is set equal to `nnpfc_patch_height_minus1 + 1`.

The variables `outPatchWidth`, `outPatchHeight`, `horCScaling`, `verCScaling`, `outPatchCWidth`, and `outPatchCHeight` are derived as follows:

$$\text{outPatchWidth} = (\text{nnpfc_pic_width_in_luma_samples} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (86)$$

$$\text{outPatchHeight} = (\text{nnpfc_pic_height_in_luma_samples} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (87)$$

$$\text{outPatchWidth} = (\text{nnpfcOutputPicWidth} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (86)$$

$$\text{outPatchHeight} = (\text{nnpfcOutputPicHeight} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (87)$$

$$\text{horCScaling} = \text{SubWidthC} / \text{outSubWidthC} \quad (88)$$

$$\text{verCScaling} = \text{SubHeightC} / \text{outSubHeightC} \quad (89)$$

$$\text{outPatchCWidth} = \text{outPatchWidth} * \text{horCScaling} \quad (90)$$

$$\text{outPatchCHeight} = \text{outPatchHeight} * \text{verCScaling} \quad (91)$$

It is a requirement of bitstream conformance that `outPatchWidth * CroppedWidth` shall be equal to `nnpfc_pic_width_in_luma_samples nnpfcOutputPicWidth * inpPatchWidth` and `outPatchHeight * CroppedHeight` shall be equal to `nnpfc_pic_height_in_luma_samples nnpfcOutputPicHeight * inpPatchHeight`.

nnpfc_padding_type indicates the process of padding when referencing sample locations outside the boundaries of the cropped decoded output picture as described in Table 23. The value of nnpfc_padding_type shall be in the range of 0 to 15, inclusive.

Table 23 – Informative description of nnpfc_padding_type values

nnpfc_padding_type	Description
0	zero padding
1	replication padding
2	reflection padding
3	wrap-around padding
4	fixed padding
5..15	Reserved

5

nnpfc_luma_padding_val indicates the luma value to be used for padding when nnpfc_padding_type is equal to 4.

nnpfc_cb_padding_val indicates the Cb value to be used for padding when nnpfc_padding_type is equal to 4.

10 nnpfc_cr_padding_val indicates the Cr value to be used for padding when nnpfc_padding_type is equal to 4.

The function InpSampleVal(y, x, picHeight, picWidth, croppedPic) with inputs being a vertical sample location y, a horizontal sample location x, a picture height picHeight, a picture width picWidth, and sample array croppedPic returns the value of sampleVal derived as

15 follows:

NOTE 6 – For the inputs to the function InpSampleVal(), the vertical location is listed before the horizontal location for compatibility with input tensor conventions of some inference engines.

```

    if( nnpfc_padding_type == 0 )
20       if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
           sampleVal = 0
       else
           sampleVal = croppedPic[ x ][ y ]
25       else if( nnpfc_padding_type == 1 )
           sampleVal = croppedPic[ Clip3( 0, picWidth - 1, x ) ][ Clip3( 0, picHeight - 1,
           y ) ]

```

(92)

```

else if( nnpfc_padding_type == 2 )
    sampleVal = croppedPic[ Reflect( picWidth - 1, x ) ][ Reflect( picHeight - 1,
y ) ]
else if( nnpfc_padding_type == 3 )
5    if( y >= 0 && y < picHeight )
        sampleVal = croppedPic[ Wrap( picWidth - 1, x ) ][ y ]
else if( nnpfc_padding_type == 4 )
    if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
        sampleVal[ 0 ] = nnpfc_luma_padding_val
10        sampleVal[ 1 ] = nnpfc_cb_padding_val
        sampleVal[ 2 ] = nnpfc_cr_padding_val
    else
        sampleVal = croppedPic[ x ][ y ]

```

The following example process may be used, with the NNPF PostProcessingFilter(), to
15 generate, in a patch-wise manner, the filtered and/or interpolated picture(s), which contain Y,
Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as
indicated by nnpfc_out_order_idc:

```

if( nnpfc_inp_order_idc == 0 || nnpfc_inp_order_idc == 2 )
    for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight )
20    for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth ) {
        DeriveInputTensors( )
        outputTensor = PostProcessingFilter( inputTensor )
        StoreOutputTensors( )
    }
25 else if( nnpfc_inp_order_idc == 1 )
    for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop +=
inpPatchHeight )
        for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft +=
inpPatchWidth ) {
30        DeriveInputTensors( )
        outputTensor = PostProcessingFilter( inputTensor )
        StoreOutputTensors( )
    }
else if( nnpfc_inp_order_idc == 3 )

```

```

    for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight * 2 )
        for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth * 2 ) {
            DeriveInputTensors( )
            outputTensor = PostProcessingFilter( inputTensor )
5            StoreOutputTensors( )
        }

```

The order of the pictures in the stored output tensor is in output order, and the output order generated by applying the NNPF in output order is interpreted to be in output order (and not conflicting with the output order of the input pictures).

- 10 nnpfc_complexity_info_present_flag equal to 1 specifies that one or more syntax elements that indicate the complexity of the NNPF associated with the nnpfc_id are present. nnpfc_complexity_info_present_flag equal to 0 specifies that no syntax elements that indicates the complexity of the NNPF associated with the nnpfc_id are present.
- nnpfc_parameter_type_idc equal to 0 indicates that the neural network uses only integer
- 15 parameters. nnpfc_parameter_type_flag equal to 1 indicates that the neural network may use floating point or integer parameters. nnpfc_parameter_type_idc equal to 2 indicates that the neural network uses only binary parameters. nnpfc_parameter_type_idc equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore
- 20 NNPF SEI messages with nnpfc_parameter_type_idc equal to 3.
- nnpfc_log2_parameter_bit_length_minus3 equal to 0, 1, 2, and 3 indicates that the neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively. When nnpfc_parameter_type_idc is present and nnpfc_log2_parameter_bit_length_minus3 is not present the neural network does not use parameters of bit length greater than 1.
- 25 nnpfc_num_parameters_idc indicates the maximum number of neural network parameters for the NNPF in units of a power of 2 048. nnpfc_num_parameters_idc equal to 0 indicates that the maximum number of neural network parameters is unknown. The value nnpfc_num_parameters_idc shall be in the range of 0 to 52, inclusive. Values of nnpfc_num_parameters_idc greater than 52 are reserved for future use by ITU-T | ISO/IEC
- 30 and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with nnpfc_num_parameters_idc greater than 52.
- If the value of nnpfc_num_parameters_idc is greater than zero, the variable maxNumParameters is derived as follows:

$$\text{maxNumParameters} = (2\,048 \ll \text{nnpfc_num_parameters_idc}) - 1 \quad (94)$$

It is a requirement of bitstream conformance that the number of neural network parameters of the NNPF shall be less than or equal to maxNumParameters.

nnpfc_num_kmac_operations_idc greater than 0 indicates that the maximum number of

5 multiply-accumulate operations per sample of the NNPF is less than or equal to

nnpfc_num_kmac_operations_idc * 1 000. nnpfc_num_kmac_operations_idc equal to 0

indicates that the maximum number of multiply-accumulate operations of the network is

unknown. The value of nnpfc_num_kmac_operations_idc shall be in the range of 0 to $2^{32} - 2$, inclusive.

10 nnpfc_total_kilobyte_size greater than 0 indicates a total size in kilobytes required to store the uncompressed parameters for the neural network. The total size in bits is a number equal to or greater than the sum of bits used to store each parameter. nnpfc_total_kilobyte_size is the

total size in bits divided by 8 000, rounded up. nnpfc_total_kilobyte_size equal to 0 indicates that the total size required to store the parameters for the neural network is unknown. The

15 value of nnpfc_total_kilobyte_size shall be in the range of 0 to $2^{32} - 2$, inclusive.

nnpfc_reserved_zero_bit_b shall be equal to 0 in bitstreams conforming to this edition of this document. Decoders shall ignore NNPF SEI messages in which nnpfc_reserved_zero_bit_b is not equal to 0.

nnpfc_payload_byte[i] contains the i-th byte of a bitstream conforming to ISO/IEC 15938-

20 17. The byte sequence nnpfc_payload_byte[i] for all present values of i shall be a complete bitstream that conforms to ISO/IEC 15938-17.

6.4 Embodiment 4

[00198] This embodiment is for the items 1, 2, 3, 4 and all their subitems, summarized above in Section 5.

25 8.28.1 Neural-network post-filter characteristics SEI message syntax

nn_post_filter_characteristics(payloadSize) {	Descriptor
nnpfc_purpose	u(16)
nnpfc_id	ue(v)
nnpfc_mode_idc	ue(v)
if(nnpfc_mode_idc == 1) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_a	u(1)
nnpfc_tag_uri	st(v)

nnpfc_uri	st(v)
}	
nnpfc_property_present_flag	u(1)
if(nnpfc_property_present_flag) {	
nnpfc_base_flag	u(1)
/* input and output formatting */	
nnpfc_num_input_pics_minus1	ue(v)
if((nnpfc_purpose & 0x02) != 0)	
nnpfc_out_sub_c_flag	u(1)
if((nnpfc_purpose & 0x20) != 0)	
nnpfc_out_colour_format_idc	u(2)
if((nnpfc_purpose & 0x04) != 0) {	
nnpfc_hor_ratio_num_minus1	u(4)
nnpfc_hor_ratio_den_minus1	u(8)
nnpfc_ver_ratio_num_minus1	u(4)
nnpfc_ver_ratio_den_minus1	u(8)
<i>nnpfc_pic_width_in_luma_samples</i>	<i>ue(v)</i>
<i>nnpfc_pic_height_in_luma_samples</i>	<i>ue(v)</i>
}	
if((nnpfc_purpose & 0x08) != 0) {	
for(i = 0; i < nnpfc_num_input_pics_minus1; i++)	
nnpfc_interpolated_pics[i]	ue(v)
for(i = 0; i <= nnpfc_num_input_pics_minus1; i++)	
nnpfc_input_pic_output_flag[i]	u(1)
}	
nnpfc_component_last_flag	u(1)
nnpfc_inp_format_idc	ue(v)
if(nnpfc_inp_format_idc == 1) {	
nnpfc_inp_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_inp_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_inp_order_idc	ue(v)
nnpfc_auxiliary_inp_idc	ue(v)

nnpfc_separate_colour_description_present_flag	u(1)
if(nnpfc_separate_colour_description_present_flag) {	
nnpfc_colour_primaries	u(8)
nnpfc_transfer_characteristics	u(8)
nnpfc_matrix_coeffs	u(8)
}	
nnpfc_out_format_idc	ue(v)
if(nnpfc_out_format_idc == 1) {	
nnpfc_out_tensor_luma_bitdepth_minus8	ue(v)
nnpfc_out_tensor_chroma_bitdepth_minus8	ue(v)
}	
nnpfc_out_order_idc	ue(v)
nnpfc_overlap	ue(v)
nnpfc_constant_patch_size_flag	u(1)
if(nnpfc_constant_patch_size_flag) {	
nnpfc_patch_width_minus1	ue(v)
nnpfc_patch_height_minus1	ue(v)
} else {	
nnpfc_extended_patch_width_cd_delta_minus1	ue(v)
nnpfc_extended_patch_height_cd_delta_minus1	ue(v)
}	
nnpfc_padding_type	ue(v)
if(nnpfc_padding_type == 4) {	
nnpfc_luma_padding_val	ue(v)
nnpfc_cb_padding_val	ue(v)
nnpfc_cr_padding_val	ue(v)
}	
nnpfc_complexity_info_present_flag	u(1)
if(nnpfc_complexity_info_present_flag) {	
nnpfc_parameter_type_idc	u(2)
if(nnpfc_parameter_type_idc != 2)	
nnpfc_log2_parameter_bit_length_minus3	u(2)
nnpfc_num_parameters_idc	u(6)

nnpfc_num_kmac_operations_idc	ue(v)
nnpfc_total_kilobyte_size	ue(v)
}	
}	
/* ISO/IEC 15938-17 bitstream */	
if(nnpfc_mode_idc == 0) {	
while(!byte_aligned())	
nnpfc_reserved_zero_bit_b	u(1)
for(i = 0; more_data_in_payload(); i++)	
nnpfc_payload_byte[i]	b(8)
}	
}	

8.28.2 Neural-network post-filter characteristics SEI message semantics

The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

Use of this SEI message requires the definition of the following variables:

- Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
- 10 – Luma sample array CroppedYPic[idx] and chroma sample arrays CroppedCbPic[idx] and CroppedCrPic[idx], when present, of the input pictures with index idx in the range of 0 to numInputPics – 1, inclusive, that are used as input for the NNPF.
- Bit depth BitDepth_Y for the luma sample array of the input pictures.
- Bit depth BitDepth_C for the chroma sample arrays, if any, of the input pictures.
- 15 – A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause 7.3.
- When nnpfc_auxiliary_inp_idc is equal to 1, a filtering strength control value StrengthControlVal that shall be a real number in the range of 0 to 1, inclusive.

Input picture with index 0 corresponds to the picture for which the NNPF defined by this NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in the range of 1 to numInputPics – 1, inclusive, precedes the input picture with index i – 1 in

output order.

When `nnpfc_purpose & 0x08` is not equal to 0 and the input picture with index 0 is associated with a frame packing arrangement SEI message with `fp_arrangement_type` equal to 5, all input pictures are associated with a frame packing arrangement SEI message with

5 `fp_arrangement_type` equal to 5 and the same value of `fp_current_frame_is_frame0_flag`.

The variables `SubWidthC` and `SubHeightC` are derived from `ChromaFormatIdc` as specified by Table 2.

NOTE 1 – More than one NNPF SEI message can be present for the same picture. When more than one NNPF SEI message with different values of `nnpfc_id` is present or

10 activated for the same picture, they can have the same or different values of `nnpfc_purpose` and `nnpfc_mode_idc`.

`nnpfc_purpose` indicates the purpose of the NNPF as specified in Table 20.

The value of `nnpfc_purpose` shall be in the range of 0 to 63, inclusive, in bitstreams conforming to this edition of this document. Values of 64 to 65 535, inclusive, for

15 `nnpfc_purpose` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_purpose` in the range of 64 to 65 535, inclusive.

Table 20 – Definition of `nnpfc_purpose`

Value	Interpretation
<code>nnpfc_purpose == 0</code>	May be used as determined by the application
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x01) == 0</code>	No general visual quality improvement
<code>(nnpfc_purpose & 0x01) != 0</code>	With general visual quality improvement
<code>nnpfc_purpose > 0 && (nnpfc_purpose & 0x02) == 0</code>	No chroma up resampling (from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format, or from the 4:4:4 chroma format to the 4:2:2 or 4:2:0 chroma format, or from the 4:2:2 chroma format to the 4:2:0 chroma format)
<code>(nnpfc_purpose & 0x02) != 0</code>	With chroma up resampling

$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x04) == 0$	No resolution upresampling (increasing or decreasing the width or height)
$(\text{nnpfc_purpose} \ \& \ 0x04) != 0$	With resolution upresampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x08) == 0$	No picture rate upresampling
$(\text{nnpfc_purpose} \ \& \ 0x08) != 0$	With picture rate upresampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x10) == 0$	No bit depth upresampling (increasing or decreasing the luma bit depth or the chroma bit depth)
$(\text{nnpfc_purpose} \ \& \ 0x10) != 0$	With bit depth upresampling
$\text{nnpfc_purpose} > 0 \ \&\& \ (\text{nnpfc_purpose} \ \& \ 0x20) == 0$	No colourization (from the 4:0:0 chroma format to the 4:2:0, 4:2:2, or 4:4:4 chroma format)
$(\text{nnpfc_purpose} \ \& \ 0x20) != 0$	With colourization

NOTE 2– When a reserved value of `nnpfc_purpose` is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by `nnpfc_purpose` being equal to that value.

5 When ChromaFormatIdc is equal to 3, `nnpfc_purpose & 0x02` shall be equal to 0.

When `ChromaFormatIdc` or `nnpfc_purpose & 0x02` is not equal to 0, `nnpfc_purpose & 0x20` shall be equal to 0.

`nnpfc_id` contains an identifying number that may be used to identify an NNPF. The value of `nnpfc_id` shall be in the range of 0 to $2^{32} - 2$, inclusive. Values of `nnpfc_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPF SEI message with `nnpfc_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore the SEI message.

When an NNPF SEI message is the first NNPF SEI message, in decoding order, that has a particular `nnpfc_id` value within the current CLVS, the following applies:

- This SEI message specifies a base NNPF.
- This SEI message pertains to the current decoded picture and all subsequent decoded

5 pictures of the current layer, in output order, until the end of the current CLVS.

...

`nnpfc_out_colour_format_idc`, when `nnpfc_purpose & 0x20` is not equal to 0, specifies the colour format of the NNPF output and consequently the values of the variables

`outSubWidthC` and `outSubHeightC`. `nnpfc_out_colour_format_idc` equal to 1 specifies that

10 the colour format of the NNPF output is the 4:2:0 format and `outSubWidthC` and

`outSubHeightC` are both equal to 2. `nnpfc_out_colour_format_idc` equal to 2 specifies that the colour format of the NNPF output is the 4:2:2 format and `outSubWidthC` is equal to 2 and

`outSubHeightC` is equal to 1. `nnpfc_out_colour_format_idc` equal to 3 specifies that the

colour format of the NNPF output is the 4:2:4 format and `outSubWidthC` and `outSubHeightC`

15 are both equal to 1. The value of `nnpfc_out_colour_format_idc` shall not be equal to 0.

When `nnpfc_purpose & 0x02` and `nnpfc_purpose & 0x20` are both equal to 0, `outSubWidthC` and `outSubHeightC` are inferred to be equal to `SubWidthC` and `SubHeightC`, respectively.

`nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` specify the width and height, respectively, of the luma sample array of the picture resulting from

20 *applying the NNPF identified by `nnpfc_id` to a cropped decoded output picture. When*

`nnpfc_pic_width_in_luma_samples` and `nnpfc_pic_height_in_luma_samples` are not present, they are inferred to be equal to `CroppedWidth` and `CroppedHeight`, respectively.

The value of `nnpfc_pic_width_in_luma_samples` shall be in the range of `CroppedWidth` to `CroppedWidth * 16`, inclusive. The value of `nnpfc_pic_height_in_luma_samples` shall be

25 *in the range of `CroppedHeight >> 4` to `CroppedHeight * 16`, inclusive.*

`nnpfc_hor_ratio_num_minus1` and `nnpfc_hor_ratio_den_minus1` specify the horizontal luma scaling ratio `nnpfcHorScalingRatio` as follows:

`nnpfcHorScalingRatio` =

`nnpfc_hor_ratio_num_minus1 + ( nnpfc_hor_ratio_den_minus1 + 1 ) ÷ 256 (X)`

30 The value of `nnpfcHorScalingRatio` shall be in the range of 1/16 to 16, inclusive. When

`nnpfc_hor_ratio_num_minus1` and `nnpfc_hor_ratio_den_minus1` are not present, the value of `nnpfcHorScalingRatio` is inferred to be equal to 1.

The value of the variable `nnpfcOutputPicWidth`, representing the width of the luma sample array of the picture(s) resulting from applying the NNPF identified by `nnpfc_id`

to a cropped decoded output picture, is derived as follows:

$$\text{nnpfcOutputPicWidth} = \text{CroppedWidth} * \text{nnpfcHorScalingRatio} \quad (\text{X})$$

`nnpfc_ver_ratio_num_minus1` and `nnpfc_ver_ratio_den_minus1` specify the vertical luma scaling ratio `nnpfcVerScalingRatio` is derived as follows:

$$\begin{aligned} & \text{nnpfcVerScalingRatio} = \\ & \text{nnpfc_ver_ratio_num_minus1} + (\text{nnpfc_ver_ratio_den_minus1} + 1) \div 256 \quad (\text{X}) \end{aligned}$$

The value of `nnpfcVerScalingRatio` shall be in the range of 1/16 to 16, inclusive. When `nnpfc_ver_ratio_num_minus1` and `nnpfc_ver_ratio_den_minus1` are not present, the value of `nnpfcVerScalingRatio` is inferred to be equal to 1.

- 10 The value of the variable `nnpfcOutputPicHeight`, representing the height of the luma sample array of the picture(s) resulting from applying the NNPF identified by `nnpfc_id` to a cropped decoded output picture, is derived as follows:

$$\text{nnpfcOutputPicHeight} = \text{CroppedHeight} * \text{nnpfcHorScalingRatio} \quad (\text{X})$$

When `nnpfc_purpose & 0x04` is not equal to 0, at least one of the following conditions shall be true:

- `nnpfcOutputPicWidth` is greater than `CroppedWidth` and `nnpfcOutputPicHeight` is not less than `CroppedHeight`, or `nnpfcOutputPicHeight` is greater than `CroppedHeight` and `nnpfcOutputPicWidth` is not less than `CroppedWidth`.
- `nnpfcOutputPicWidth` is less than `CroppedWidth` and `nnpfcOutputPicHeight` is not greater than `CroppedHeight`, or `nnpfcOutputPicHeight` is less than `CroppedHeight` and `nnpfcOutputPicWidth` is not greater than `CroppedWidth`.

`nnpfc_num_input_pics_minus1` plus 1 specifies the number of decoded output pictures used as input for the NNPF. The value of `nnpfc_num_input_pics_minus1` shall be in the range of 0 to 63, inclusive. When `nnpfc_purpose & 0x08` is not equal to 0, the value of

- 25 `nnpfc_num_input_pics_minus1` shall be greater than 0.

...

`nnpfc_out_format_idc` equal to 0 indicates that the sample values output by the NNPF are real numbers where the value range of 0 to 1, inclusive, maps linearly to the unsigned integer value range of 0 to $(1 \ll \text{bitDepth}) - 1$, inclusive, for any desired bit depth `bitDepth` for subsequent post-processing or displaying.

- 30 `nnpfc_out_format_idc` equal to 1 indicates that the luma sample values output by the NNPF are unsigned integer numbers in the range of 0 to $(1 \ll (\text{nnpfc_out_tensor_luma_bitdepth_minus8} + 8)) - 1$, inclusive, and the chroma sample values output by the NNPF are unsigned integer numbers in the range of 0 to

$(1 \ll (\text{nnpfc_out_tensor_chroma_bitdepth_minus8} + 8)) - 1$, inclusive.

Values of `nnpfc_out_format_idc` greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document.

Decoders conforming to this edition of this document shall ignore NNPF SEI messages that

5 contain reserved values of `nnpfc_out_format_idc`.

`nnpfc_out_tensor_luma_bitdepth_minus8` plus 8 specifies the bit depth of luma sample values in the output integer tensor. The value of `nnpfc_out_tensor_luma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

10 `nnpfc_out_tensor_chroma_bitdepth_minus8` plus 8 specifies the bit depth of chroma sample values in the output integer tensor. The value of `nnpfc_out_tensor_chroma_bitdepth_minus8` shall be in the range of 0 to 24, inclusive.

When `nnpfc_purpose & 0x10` is not equal to 0, the value of `nnpfc_out_format_idc` shall be equal to 1 and at least one of the following conditions shall be true:

- `(nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is greater than `BitDepthY` **and**
- 15 **`nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is not less than `BitDepthC`), or**
`(nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is greater than `BitDepthC` **and**
`nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is not less than `BitDepthY`).
- `(nnpfc_out_tensor_luma_bitdepth_minus8 + 8` is less than `BitDepthY` **and**
- `nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is not greater than `BitDepthC`), or**
- 20 **`(nnpfc_out_tensor_chroma_bitdepth_minus8 + 8` is less than `BitDepthC` **and****
`nnpfc_out_tensor_luma_bitdepth_minus1 + 1` is not greater than `BitDepthY`).

`nnpfc_out_order_idc` indicates the output order of samples resulting from the NNPF.

The value of `nnpfc_out_order_idc` shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this edition of this document. Values of 4 to 255, inclusive, for

25 `nnpfc_out_order_idc` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_out_order_idc` in the range of 4 to 255, inclusive. Values of `nnpfc_out_order_idc` greater than 255 shall not be present in bitstreams conforming to this edition of this document and are not reserved for future use.

30 When `nnpfc_purpose & 0x02` is not equal to 0, `nnpfc_out_order_idc` shall not be equal to 3.

Table 22 contains an informative description of `nnpfc_out_order_idc` values.

Table 22 – Description of `nnpfc_out_order_idc` values

<code>nnpfc_out_</code>	Description
-------------------------	-------------

order_idc	
0	Only the luma matrix is present in the output tensor, thus the number of channels is 1.
1	Only the chroma matrices are present in the output tensor, thus the number of channels is 2.
2	The luma and chroma matrices are present in the output tensor, thus the number of channels is 3.
3	Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This nnpfc_out_order_idc can only be used when the output chroma format is 4:2:0.
4..255	Reserved

The process StoreOutputTensors(), for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensor outputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensor, is specified as follows:

```

for( i = 0; i < numOutputPics; i++ ) {
    if( nnpfc_out_order_idc == 0 )
        for( yP = 0; yP < outPatchHeight; yP++ )
10      for( xP = 0; xP < outPatchWidth; xP++ ) {
            yY = cTop * outPatchHeight / inpPatchHeight + yP
            xY = cLeft * outPatchWidth / inpPatchWidth + xP
            if ( yY < nnpfc_pic_height_in_luma_samples && xY <
    nnpfc_pic_width_in_luma_samples )
15      if ( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth )
                if( !nnpfc_component_last_flag )
                    FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
                else
                    FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
20      }
    else if( nnpfc_out_order_idc == 1 )
        for( yP = 0; yP < outPatchCHeight; yP++ )

```

(85)

```

for( xP = 0; xP < outPatchCWidth; xP++ ) {
    xSrc = cLeft * horCScaling + xP
    ySrc = cTop * verCScaling + yP
    if ( ySrc < nnpfc_pic_height_in_luma_samples / outSubHeightC &&
5      xSrc < nnpfc_pic_width_in_luma_samples / outSubWidthC )
    if ( ySrc < nnpfcOutputPicHeight / outSubHeightC &&
        xSrc < nnpfcOutputPicWidth / outSubWidthC )
        if( !nnpfc_component_last_flag ) {
            FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
10          FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
        } else {
            FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
            FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
        }
15      }
else if( nnpfc_out_order_idc == 2 )
    for( yP = 0; yP < outPatchHeight; yP++ )
        for( xP = 0; xP < outPatchWidth; xP++ ) {
            yY = cTop * outPatchHeight / inpPatchHeight + yP
            xY = cLeft * outPatchWidth / inpPatchWidth + xP
20          yC = yY / outSubHeightC
            xC = xY / outSubWidthC
            yPc = ( yP / outSubHeightC ) * outSubHeightC
            xPc = ( xP / outSubWidthC ) * outSubWidthC
25          if ( yY < nnpfc_pic_height_in_luma_samples && xY <
nnpfc_pic_width_in_luma_samples )
            if ( yY < nnpfcOutputPicHeight && xY < nnpfcOutputPicWidth )
                if( !nnpfc_component_last_flag ) {
                    FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
30                  FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 1 ][ yPc ][ xPc ]
                    FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ 2 ][ yPc ][ xPc ]
                } else {
                    FilteredYPic[ i ][ xY ][ yY ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
                    FilteredCbPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 1 ]

```

```

        FilteredCrPic[ i ][ xC ][ yC ] = outputTensor[ 0 ][ i ][ yPc ][ xPc ][ 2 ]
    }
}
else if( nnpfc_out_order_idc == 3 )
5   for( yP = 0; yP < outPatchHeight; yP++ )
    for( xP = 0; xP < outPatchWidth; xP++ ) {
        ySrc = cTop / 2 * outPatchHeight / inpPatchHeight + yP
        xSrc = cLeft / 2 * outPatchWidth / inpPatchWidth + xP
        if ( ySrc < nnpfc_pic_height_in_luma_samples / 2 &&
10      xSrc < nnpfc_pic_width_in_luma_samples / 2 )
        if ( ySrc < nnpfcOutputPicHeight / 2 &&
          xSrc < nnpfcOutputPicWidth / 2 )
            if( !nnpfc_component_last_flag ) {
                FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
15      outputTensor[ 0 ][ i ][ 0 ][ yP ][ xP ]
                FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
                outputTensor[ 0 ][ i ][ 1 ][ yP ][ xP ]
                FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
                outputTensor[ 0 ][ i ][ 2 ][ yP ][ xP ]
20      FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =
                outputTensor[ 0 ][ i ][ 3 ][ yP ][ xP ]
                FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 4 ][ yP ][ xP ]
                FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ 5 ][ yP ][ xP ]
            } else {
25      FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 ] =
                outputTensor[ 0 ][ i ][ yP ][ xP ][ 0 ]
                FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 ] =
                outputTensor[ 0 ][ i ][ yP ][ xP ][ 1 ]
                FilteredYPic[ i ][ xSrc * 2 ][ ySrc * 2 + 1 ] =
30      outputTensor[ 0 ][ i ][ yP ][ xP ][ 2 ]
                FilteredYPic[ i ][ xSrc * 2 + 1 ][ ySrc * 2 + 1 ] =
                outputTensor[ 0 ][ i ][ yP ][ xP ][ 3 ]
                FilteredCbPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 4 ]
                FilteredCrPic[ i ][ xSrc ][ ySrc ] = outputTensor[ 0 ][ i ][ yP ][ xP ][ 5 ]

```

```

    }
  }
}

```

nnpfc_overlap indicates the overlapping horizontal and vertical sample counts of adjacent

5 input tensors of the NNPF. The value of nnpfc_overlap shall be in the range of 0 to 16 383, inclusive.

nnpfc_constant_patch_size_flag equal to 1 indicates that the NNPF accepts exactly the patch size indicated by nnpfc_patch_width_minus1 and nnpfc_patch_height_minus1 as input.

nnpfc_constant_patch_size_flag equal to 0 indicates that the NNPF accepts as input any patch
10 size with width inpPatchWidth and height inpPatchHeight such that the width of an extended patch (i.e., a patch plus the overlapping area), which is equal to

inpPatchWidth + 2 * nnpfc_overlap, is a positive integer multiple of

nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap, and the height of the extended patch, which is equal to inpPatchHeight + 2 * nnpfc_overlap, is a positive integer

15 multiple of nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap.

nnpfc_patch_width_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to 1, indicates the horizontal sample counts of the patch size required for the input to the NNPF.

The value of nnpfc_patch_width_minus1 shall be in the range of 0 to Min(32 766, CroppedWidth - 1), inclusive.

20 nnpfc_patch_height_minus1 plus 1, when nnpfc_constant_patch_size_flag equal to 1, indicates the vertical sample counts of the patch size required for the input to the NNPF. The value of nnpfc_patch_height_minus1 shall be in the range of 0 to Min(32 766, CroppedHeight - 1), inclusive.

nnpfc_extended_patch_width_cd_delta_minus1 plus 1 plus 2 * nnpfc_overlap, when

25 nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the width of an extended patch required for the input to the NNPF. The value of nnpfc_extended_patch_width_cd_delta_minus1 shall be in the range of 0 to Min(32 766, CroppedWidth - 1), inclusive.

nnpfc_extended_patch_height_cd_delta_minus1 plus 1 plus 2 * nnpfc_overlap, when

30 nnpfc_constant_patch_size_flag equal to 0, indicates a common divisor of all allowed values of the height of an extended patch required for the input to the NNPF. The value of nnpfc_extended_patch_height_cd_delta_minus1 shall be in the range of 0 to Min(32 766, CroppedHeight - 1), inclusive.

Let the variables inpPatchWidth and inpPatchHeight be the patch size width and the patch

size height, respectively.

If `nnpfc_constant_patch_size_flag` is equal to 0, the following applies:

- The values of `inpPatchWidth` and `inpPatchHeight` are either provided by external means not specified in this document or set by the post-processor itself.
- 5 – The value of `inpPatchWidth + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_width_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and `inpPatchWidth` shall be less than or equal to `CroppedWidth`. The value of `inpPatchHeight + 2 * nnpfc_overlap` shall be a positive integer multiple of `nnpfc_extended_patch_height_cd_delta_minus1 + 1 + 2 * nnpfc_overlap` and
- 10 `inpPatchHeight` shall be less than or equal to `CroppedHeight`.

Otherwise (`nnpfc_constant_patch_size_flag` is equal to 1), the value of `inpPatchWidth` is set equal to `nnpfc_patch_width_minus1 + 1` and the value of `inpPatchHeight` is set equal to `nnpfc_patch_height_minus1 + 1`.

- The variables `outPatchWidth`, `outPatchHeight`, `horCScaling`, `verCScaling`, `outPatchCWidth`,
 15 and `outPatchCHeight` are derived as follows:

$$\text{outPatchWidth} = (\text{nnpfc_pic_width_in_luma_samples} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (86)$$

$$\text{outPatchHeight} = (\text{nnpfc_pic_height_in_luma_samples} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (87)$$

$$\text{outPatchWidth} = (\text{nnpfcOutputPicWidth} * \text{inpPatchWidth}) / \text{CroppedWidth} \quad (86)$$

$$\text{outPatchHeight} = (\text{nnpfcOutputPicHeight} * \text{inpPatchHeight}) / \text{CroppedHeight} \quad (87)$$

$$\text{horCScaling} = \text{SubWidthC} / \text{outSubWidthC} \quad (88)$$

$$\text{verCScaling} = \text{SubHeightC} / \text{outSubHeightC} \quad (89)$$

$$\text{outPatchCWidth} = \text{outPatchWidth} * \text{horCScaling} \quad (90)$$

$$\text{outPatchCHeight} = \text{outPatchHeight} * \text{verCScaling} \quad (91)$$

- It is a requirement of bitstream conformance that `outPatchWidth * CroppedWidth` shall be equal to `nnpfc_pic_width_in_luma_samples nnpfcOutputPicWidth * inpPatchWidth` and
 30 `outPatchHeight * CroppedHeight` shall be equal to `nnpfc_pic_height_in_luma_samples nnpfcOutputPicHeight * inpPatchHeight`.

`nnpfc_padding_type` indicates the process of padding when referencing sample locations outside the boundaries of the cropped decoded output picture as described in Table 23. The value of `nnpfc_padding_type` shall be in the range of 0 to 15, inclusive.

Table 23 – Informative description of nnpfc_padding_type values

nnpfc_padding_type	Description
0	zero padding
1	replication padding
2	reflection padding
3	wrap-around padding
4	fixed padding
5..15	Reserved

nnpfc_luma_padding_val indicates the luma value to be used for padding when nnpfc_padding_type is equal to 4.

- 5 nnpfc_cb_padding_val indicates the Cb value to be used for padding when nnpfc_padding_type is equal to 4.

nnpfc_cr_padding_val indicates the Cr value to be used for padding when nnpfc_padding_type is equal to 4.

- The function InpSampleVal(y, x, picHeight, picWidth, croppedPic) with inputs being a
 10 vertical sample location y, a horizontal sample location x, a picture height picHeight, a picture width picWidth, and sample array croppedPic returns the value of sampleVal derived as follows:

- NOTE 6 – For the inputs to the function InpSampleVal(), the vertical location is listed before the horizontal location for compatibility with input tensor conventions of some
 15 inference engines.

```

    if( nnpfc_padding_type == 0 )
        if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
            sampleVal = 0
        else
20         sampleVal = croppedPic[ x ][ y ]
    else if( nnpfc_padding_type == 1 )
        sampleVal = croppedPic[ Clip3( 0, picWidth - 1, x ) ][ Clip3( 0, picHeight - 1,
y ) ]
    else if( nnpfc_padding_type == 2 )
25         sampleVal = croppedPic[ Reflect( picWidth - 1, x ) ][ Reflect( picHeight - 1,
y ) ]
  
```

(92)

```

else if( nnpfc_padding_type == 3 )
    if( y >= 0 && y < picHeight )
        sampleVal = croppedPic[ Wrap( picWidth - 1, x ) ][ y ]
else if( nnpfc_padding_type == 4 )
5     if( y < 0 || x < 0 || y >= picHeight || x >= picWidth )
        sampleVal[ 0 ] = nnpfc_luma_padding_val
        sampleVal[ 1 ] = nnpfc_cb_padding_val
        sampleVal[ 2 ] = nnpfc_cr_padding_val
    else
10     sampleVal = croppedPic[ x ][ y ]

```

The following example process may be used, with the NNPF PostProcessingFilter(), to generate, in a patch-wise manner, the filtered and/or interpolated picture(s), which contain Y, Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as indicated by nnpfc_out_order_idc:

```

15     if( nnpfc_inp_order_idc == 0 || nnpfc_inp_order_idc == 2 )
        for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight )
            for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth ) {
                DeriveInputTensors( )
                outputTensor = PostProcessingFilter( inputTensor )
20                StoreOutputTensors( )
            }
    else if( nnpfc_inp_order_idc == 1 )
        for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop +=
inpPatchHeight )
25        for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft +=
inpPatchWidth ) {
                                                    (93)
                DeriveInputTensors( )
                outputTensor = PostProcessingFilter( inputTensor )
                StoreOutputTensors( )
30            }
    else if( nnpfc_inp_order_idc == 3 )
        for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight * 2 )
            for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth * 2 ) {
                DeriveInputTensors( )

```

```

        outputTensor = PostProcessingFilter( inputTensor )
        StoreOutputTensors( )
    }

```

The order of the pictures in the stored output tensor is in output order, and the output order generated by applying the NNPF in output order is interpreted to be in output order (and not conflicting with the output order of the input pictures).

`nnpfc_complexity_info_present_flag` equal to 1 specifies that one or more syntax elements that indicate the complexity of the NNPF associated with the `nnpfc_id` are present.

`nnpfc_complexity_info_present_flag` equal to 0 specifies that no syntax elements that indicates the complexity of the NNPF associated with the `nnpfc_id` are present.

`nnpfc_parameter_type_idc` equal to 0 indicates that the neural network uses only integer parameters. `nnpfc_parameter_type_flag` equal to 1 indicates that the neural network may use floating point or integer parameters. `nnpfc_parameter_type_idc` equal to 2 indicates that the neural network uses only binary parameters. `nnpfc_parameter_type_idc` equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_parameter_type_idc` equal to 3.

`nnpfc_log2_parameter_bit_length_minus3` equal to 0, 1, 2, and 3 indicates that the neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively.

When `nnpfc_parameter_type_idc` is present and `nnpfc_log2_parameter_bit_length_minus3` is not present the neural network does not use parameters of bit length greater than 1.

`nnpfc_num_parameters_idc` indicates the maximum number of neural network parameters for the NNPF in units of a power of 2 048. `nnpfc_num_parameters_idc` equal to 0 indicates that the maximum number of neural network parameters is unknown. The value

`nnpfc_num_parameters_idc` shall be in the range of 0 to 52, inclusive. Values of `nnpfc_num_parameters_idc` greater than 52 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPF SEI messages with `nnpfc_num_parameters_idc` greater than 52.

If the value of `nnpfc_num_parameters_idc` is greater than zero, the variable `maxNumParameters` is derived as follows:

$$\text{maxNumParameters} = (2\,048 \ll \text{nnpfc_num_parameters_idc}) - 1 \quad (94)$$

It is a requirement of bitstream conformance that the number of neural network parameters of the NNPF shall be less than or equal to `maxNumParameters`.

nnpfc_num_kmac_operations_idc greater than 0 indicates that the maximum number of multiply-accumulate operations per sample of the NNPF is less than or equal to

nnpfc_num_kmac_operations_idc * 1 000. nnpfc_num_kmac_operations_idc equal to 0 indicates that the maximum number of multiply-accumulate operations of the network is

5 unknown. The value of nnpfc_num_kmac_operations_idc shall be in the range of 0 to $2^{32} - 2$, inclusive.

nnpfc_total_kilobyte_size greater than 0 indicates a total size in kilobytes required to store the uncompressed parameters for the neural network. The total size in bits is a number equal to or

10 greater than the sum of bits used to store each parameter. nnpfc_total_kilobyte_size is the total size in bits divided by 8 000, rounded up. nnpfc_total_kilobyte_size equal to 0 indicates that the total size required to store the parameters for the neural network is unknown. The value of nnpfc_total_kilobyte_size shall be in the range of 0 to $2^{32} - 2$, inclusive.

nnpfc_reserved_zero_bit_b shall be equal to 0 in bitstreams conforming to this edition of this document. Decoders shall ignore NNPF SEI messages in which nnpfc_reserved_zero_bit_b

15 is not equal to 0.

nnpfc_payload_byte[i] contains the i-th byte of a bitstream conforming to ISO/IEC 15938-17. The byte sequence nnpfc_payload_byte[i] for all present values of i shall be a complete bitstream that conforms to ISO/IEC 15938-17.

[0199] More details of the embodiments of the present disclosure will be described
20 below which are related to neural-network post-processing filter. As used herein, the term “neural-network post-processing filter” and “neural-network post-filter” may be used interchangeably. The embodiments of the present disclosure should be considered as examples to explain the general concepts and should not be interpreted in a narrow way. Furthermore, these embodiments can be applied individually or combined in any manner.

25 [0200] Fig. 5 illustrates a flowchart of a method 500 for video processing in accordance with some embodiments of the present disclosure. As shown in Fig. 5, at 502, a conversion between a video and a bitstream of the video is performed. In some embodiments, the conversion may include encoding the video into the bitstream. Alternatively or additionally, the conversion may include decoding the video from the
30 bitstream.

[0201] A neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video. For example, the at least one picture may be used as an input for the NNPF. In some embodiments, the at least one picture may comprise at least one

decoded picture of the video. Alternatively, the at least one picture may comprise at least one cropped decoded picture of the video. For example, the decoded picture and/or the cropped decoded picture may be outputted by a decoder that decodes the video from the bitstream. In some further embodiments, the at least one picture may comprise an output
5 of a further NNPF used to filter one or more decoded pictures or cropped decoded pictures of the video. For example, the NNPF is concatenated with the further NNPF. It should be understood that the possible implementations of the at least one picture associated with the video described here are merely illustrative and therefore should not be construed as limiting the present disclosure in any way.

10 **[0202]** The bitstream comprises a first indication indicating a purpose of the NNPF. By way of example, the first indication may be comprised in a supplemental enhancement information (SEI) message or any other suitable video message unit in the bitstream. For example, the first indication may comprise a syntax element `nnpfc_purpose`. It should be understood that the name for an indication and/or a syntax element is used only for
15 illustration rather than limitation, the indication(s) and the syntax element(s) mentioned throughout the present disclosure may be represented by any other suitable string different from that mentioned in this disclosure. The scope of the present disclosure is not limited in this respect.

[0203] In addition, a first candidate of a plurality of candidates for the purpose at least
20 indicates that decreasing a width of the at least one picture and/or decreasing a height of the at least one picture is applicable. In other words, the NNPF may support decreasing the width and/or height of the at least one picture. For example, if the first candidate is selected to be comprised in the purpose of the NNPF, the NNPF may be used to decrease the width and/or height of the at least one picture dependent on further information.

25 **[0204]** In one example embodiment, the first candidate may be resampling a resolution of the at least one picture(which may also be referred to as resolution resampling for short). For example, resolution resampling may support resolution downsampling and resolution upsampling. Resolution resampling may indicate that at least one of the following is applicable: decreasing the width of the at least one picture, decreasing the height of the at
30 least one picture, increasing the width of the at least one picture, or increasing the height of the at least one picture. In this case, a scaling ratio of the width and/or the height of the at least one picture may be in a predetermined range. By way of example rather than limitation, a lower limit of the predetermined range may be equal to 1/16. Additionally

or alternatively, an upper limit of the predetermined range may be equal to 16. For example, the scaling ration may be in a range from 1/16 to 16 inclusive. It should be understood that the specific values recited herein are intended to be exemplary rather than limiting the scope of the present disclosure. For example, the lower limit of the predetermined range may also be equal to 1, 1/2, 1/4, 1/8, or the like. Similarly, the upper limit of the predetermined range may also be equal to 1, 2, 4, 8, or the like.

[0205] In another example embodiment, the first candidate may be downsampling the resolution of the at least one picture (which may also be referred to as resolution downsampling for short). Resolution downsampling may indicate that decreasing the width and/or the height of the at least one picture is applicable. In addition, the purpose of the NNPF may be disallowed to comprise both downsampling the resolution of the at least one picture and upsampling the resolution of the at least one picture. The downsampling ratio of the width and/or the height of the at least one picture may be in a predetermined range. By way of example, a lower limit of the predetermined range may be equal to 1, 1/2, 1/4, 1/8, or 1/16. Additionally or alternatively, an upper limit of the predetermined range may be equal to 1 or the like.

[0206] In view of the above, one of candidates for a purpose of NNPF indicates that decreasing a width and/or a height of the at least one picture is applicable. Compared with the conventional solution where only resolution upsampling is supported, the proposed method can advantageously make it possible for the NNPF to support resolution downsampling. Thereby, the functionality of NNPF may be extended and enhanced.

[0207] In some embodiments, a second candidate of the plurality of candidates for the purpose may at least indicate that at least one of the following is applicable: decreasing a luma bit depth of the at least one picture, or decreasing a chroma bit depth of the at least one picture. For example, the bit depth of one or more luma and/or chroma sample values of the at least one picture may be decreased, such as, from 8 bit to 5 bit or the like.

[0208] In one example embodiment, the second candidate may be downsampling a bit depth of the at least one picture (which may also be referred to as bit depth downsampling for short). Bit depth downsampling indicates that luma bit depth and/or chroma bit depth may be decreased. In addition, the purpose of the NNPF may be disallowed to comprise downsampling the bit depth of the at least one picture and upsampling the bit depth of the at least one picture.

[0209] In another example embodiment, the second candidate may be resampling a bit depth of the at least one picture (which may also be referred to as bit depth resampling for short). For example, the bit depth resampling may support bit depth downsampling and bit depth upsampling. Bit depth resampling may indicate that at least one of the following is applicable: decreasing the luma bit depth of the at least one picture, decreasing the chroma bit depth of the at least one picture, increasing the luma bit depth of the at least one picture, or increasing the chroma bit depth of the at least one picture.

[0210] In view of the above, one of candidates for a purpose of NNPF indicates that decreasing a luma bit depth or a chroma bit depth of the at least one picture is applicable.

Compared with the conventional solution where only bit depth upsampling is supported, the proposed method can advantageously make it possible for the NNPF to support bit depth downsampling. Thereby, the functionality of NNPF may be further extended and enhanced.

[0211] In some embodiments, a third candidate of the plurality of candidates for the purpose may at least indicate that downsampling a chroma format of the at least one picture is applicable. In one example embodiment, the third candidate may be downsampling the chroma format of the at least one picture (which may also be referred to as chroma format downsampling or chroma downsampling for short). In addition, the purpose of the NNPF may be disallowed to comprise downsampling the chroma format of the at least one picture and upsampling the chroma format of the at least one picture. By way of example, the chroma format downsampling may comprise changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format. Additionally or alternatively, the chroma format downsampling may comprise changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format, or changing the chroma format from 4:2:0 chroma format to 4:0:0 chroma format. It should be understood that the above examples are described merely for purpose of description. The scope of the present disclosure is not limited in this respect.

[0212] In another example embodiment, the third candidate may be resampling the chroma format of the at least one picture (which may also be referred to as chroma format resampling or chroma resampling for short). For example, the chroma format resampling may support chroma format downsampling and chroma format upsampling. By way of

example, the chroma format resampling may comprise changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format, changing the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format.

[0213] Additionally or alternatively, the chroma format resampling may comprise at least one of the following: changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format, changing the chroma format from 4:0:0 chroma format to 4:2:0 chroma format, changing the chroma format from 4:0:0 chroma format to 4:2:2 chroma format, changing the chroma format from 4:0:0 chroma format to 4:4:4 chroma format, changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format, changing the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format. It should be understood that the above examples are described merely for purpose of description. The scope of the present disclosure is not limited in this respect.

[0214] In view of the above, one of candidates for a purpose of NNPF indicates that downsampling a chroma format of the at least one picture is applicable. Compared with the conventional solution where only chroma format upsampling is supported, the proposed method can advantageously make it possible for the NNPF to support chroma format downsampling. Thereby, the functionality of NNPF may be further extended and enhanced.

[0215] In some embodiments, a fourth candidate of the plurality of candidates for the purpose may at least indicate that downsampling a picture rate of the video is applicable. As used herein, the picture rate may refer to a frame rate of the video.

[0216] In one example embodiment, the fourth candidate may be downsampling the picture rate of the video (which may also be referred to as picture rate downsampling for short). In addition, the purpose of the NNPF may be disallowed to comprise

downsampling the picture rate of the video and upsampling the picture rate of the video. By way of example rather than limitation, the picture rate of the video may be decreased from 60 frame per second (fps) to 30 fps.

[0217] In another example embodiment, the fourth candidate may be resampling the picture rate of the video (which may also be referred to as picture rate resampling for short). For example, the picture rate resampling may support picture rate downsampling and picture rate upsampling. In other words, the picture rate resampling may indicate that downsampling the picture rate of the video or upsampling the picture rate of the video is applicable.

[0218] In view of the above, one of candidates for a purpose of NNPF indicates that downsampling a picture rate of the video is applicable. Compared with the conventional solution where only picture rate upsampling is supported, the proposed method can advantageously make it possible for the NNPF to support picture rate downsampling. Thereby, the functionality of NNPF may be further extended and enhanced.

[0219] In some embodiments, the above-mentioned purpose candidates may be combined based on a bit masking method. For example, each of the plurality of candidates for the purpose may correspond to a bit mask, and the bit mask may be used for determining the purpose of the NNPF based on the first indication. In addition, one bit in the first indication indicates whether the purpose of the NNPF may comprise one of the plurality of candidates for the purpose. By way of example rather than limitation, if a value of a first bit (such as bit 0, bit 1 or the like) in the first indication is equal to a first value (such as 1 or the like), the purpose of the NNPF may comprise the first candidate for the purpose. If the value of the first bit is equal to a second value (such as 0 or the like), the purpose of the NNPF may do not comprise the first candidate for the purpose.

[0220] Furthermore, the purpose of the NNPF may be determined based on a result of applying a bitwise operation on the first indication and at least one bit mask. By way of example rather than limitation, bit 0 may represent the least significant bit in a syntax element `nnpfc_purpose`, while bit 1 in the syntax element `nnpfc_purpose` may correspond to resolution resampling. A bit mask 0x04 may be used for determining whether the purpose of the NNPF comprises resolution resampling. If (`nnpfc_purpose & 0x04`) is equal to 0, the purpose of the NNPF does not comprise resolution resampling. If (`nnpfc_purpose & 0x04`) is larger than 0, the purpose of the NNPF comprises resolution

resampling. The operator “&” represents a bitwise AND operation. It should be understood that the above illustrations are described merely for purpose of description. The scope of the present disclosure is not limited in this respect.

[0221] In view of the above, one of the plurality of candidates for the purpose of the NNPF corresponds to a bit mask that is used for determining the purpose of the NNPF based on the first indication. Compared with the conventional solution, the proposed method advantageously provides a systematic scheme for signaling the purpose of the NNPF, and thus supports a possible extension of the purpose. Thereby, potential instability and logical issues can be avoided, and the coding efficiency can be improved.

[0222] According to further embodiments of the present disclosure, a non-transitory computer-readable recording medium is provided. The non-transitory computer-readable recording medium stores a bitstream of a video which is generated by a method performed by an apparatus for video processing. In the method, a conversion between a video and a bitstream of the video is performed. A neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video. The bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture.

[0223] According to still further embodiments of the present disclosure, a method for storing bitstream of a video is provided. In the method, a conversion between a video and a bitstream of the video is performed. A neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video. The bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture. Moreover, the bitstream is stored in a non-transitory computer-readable recording medium.

[0224] Implementations of the present disclosure can be described in view of the following clauses, the features of which can be combined in any reasonable manner.

[0225] Clause 1. A method for video processing, comprising: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing

filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the

5 at least one picture.

[0226] Clause 2. The method of clause 1, wherein the first indication comprises a syntax element `nnpfc_purpose`.

[0227] Clause 3. The method of any of clauses 1-2, wherein the at least one picture comprises at least one decoded picture or at least one cropped decoded picture of the video.

10 [0228] Clause 4. The method of any of clauses 1-3, wherein the first candidate is resampling a resolution of the at least one picture.

[0229] Clause 5. The method of clause 4, wherein resampling the resolution of the at least one picture indicates that at least one of the following is applicable: decreasing the width of the at least one picture, decreasing the height of the at least one picture, 15 increasing the width of the at least one picture, or increasing the height of the at least one picture.

[0230] Clause 6. The method of any of clauses 1-5, wherein a scaling ratio of at least one of the width or the height of the at least one picture is in a predetermined range.

[0231] Clause 7. The method of clause 6, wherein a lower limit of the predetermined 20 range is equal to 1/16.

[0232] Clause 8. The method of any of clauses 6-7, wherein an upper limit of the predetermined range is equal to 16.

[0233] Clause 9. The method of clause 6 or 8, wherein a lower limit of the predetermined range is equal to one of the following: 1, 1/2, 1/4, or 1/8.

25 [0234] Clause 10. The method of any of clauses 6-7 and 9, wherein an upper limit of the predetermined range is equal to one of the following: 1, 2, 4, or 8.

[0235] Clause 11. The method of any of clauses 1-3, wherein the first candidate is downsampling a resolution of the at least one picture.

[0236] Clause 12. The method of clause 11, wherein the purpose of the NNPF is 30 disallowed to comprise downsampling the resolution of the at least one picture and

upsampling the resolution of the at least one picture.

[0237] Clause 13. The method of any of clauses 11-12, wherein a downsampling ratio of at least one of the width or the height of the at least one picture is in a predetermined range.

5 [0238] Clause 14. The method of clause 13, wherein a lower limit of the predetermined range is equal to one of the following: 1, 1/2, 1/4, 1/8, or 1/16.

[0239] Clause 15. The method of any of clauses 13-14, wherein an upper limit of the predetermined range is equal to 1.

10 [0240] Clause 16. The method of any of clauses 1-15, wherein a second candidate of the plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a luma bit depth of the at least one picture, or decreasing a chroma bit depth of the at least one picture.

[0241] Clause 17. The method of clause 16, wherein the second candidate is downsampling a bit depth of the at least one picture.

15 [0242] Clause 18. The method of clause 17, wherein the purpose of the NNPF is disallowed to comprise downsampling the bit depth of the at least one picture and upsampling the bit depth of the at least one picture.

[0243] Clause 19. The method of clause 16, wherein the second candidate is resampling a bit depth of the at least one picture.

20 [0244] Clause 20. The method of clause 19, wherein resampling the bit depth of the at least one picture indicates that at least one of the following is applicable: decreasing the luma bit depth of the at least one picture, decreasing the chroma bit depth of the at least one picture, increasing the luma bit depth of the at least one picture, or increasing the chroma bit depth of the at least one picture.

25 [0245] Clause 21. The method of any of clauses 1-20, wherein a third candidate of the plurality of candidates for the purpose at least indicates that downsampling a chroma format of the at least one picture is applicable.

[0246] Clause 22. The method of clause 21, wherein the third candidate is downsampling the chroma format of the at least one picture.

30 [0247] Clause 23. The method of clause 22, wherein the purpose of the NNPF is

disallowed to comprise downsampling the chroma format of the at least one picture and upsampling the chroma format of the at least one picture.

[0248] Clause 24. The method of any of clauses 22-23, wherein downsampling the chroma format of the at least one picture comprises at least one of the following: changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format.

[0249] Clause 25. The method of any of clauses 22-23, wherein downsampling the chroma format of the at least one picture comprises at least one of the following: changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format, or changing the chroma format from 4:2:0 chroma format to 4:0:0 chroma format.

[0250] Clause 26. The method of clause 21, wherein the third candidate is resampling the chroma format of the at least one picture.

[0251] Clause 27. The method of clause 26, wherein resampling the chroma format of the at least one picture comprises at least one of the following: changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format, changing the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format.

[0252] Clause 28. The method of clause 26, wherein resampling the chroma format of the at least one picture comprises at least one of the following: changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format, changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format, changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format, changing the chroma format from 4:0:0 chroma format to 4:2:0 chroma format, changing the chroma format from 4:0:0 chroma format to 4:2:2 chroma format, changing the chroma format from 4:0:0 chroma format to 4:4:4 chroma format, changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format, changing

the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format.

[0253] Clause 29. The method of any of clauses 1-28, wherein a fourth candidate of the plurality of candidates for the purpose at least indicates that downsampling a picture rate of the video is applicable.

[0254] Clause 30. The method of clause 29, wherein the fourth candidate is downsampling the picture rate of the video.

[0255] Clause 31. The method of clause 30, wherein the purpose of the NNPF is disallowed to comprise downsampling the picture rate of the video and upsampling the picture rate of the video.

[0256] Clause 32. The method of clause 29, wherein the fourth candidate is resampling the picture rate of the video.

[0257] Clause 33. The method of clause 32, wherein resampling the picture rate of the video indicates that downsampling the picture rate of the video or upsampling the picture rate of the video is applicable.

[0258] Clause 34. The method of any of clauses 1-33, wherein the conversion includes encoding the video into the bitstream.

[0259] Clause 35. The method of any of clauses 1-33, wherein the conversion includes decoding the video from the bitstream.

[0260] Clause 36. An apparatus for video processing comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method in accordance with any of clauses 1-35.

[0261] Clause 37. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of clauses 1-35.

[0262] Clause 38. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by an apparatus for video processing, wherein the method comprises: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is

applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one

5 picture.

[0263] Clause 39. A method for storing a bitstream of a video, comprising: performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first
10 candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture; and storing the bitstream in a non-transitory computer-readable recording medium.

Example Device

15 [0264] Fig. 6 illustrates a block diagram of a computing device 600 in which various embodiments of the present disclosure can be implemented. The computing device 600 may be implemented as or included in the source device 110 (or the video encoder 114 or 200) or the destination device 120 (or the video decoder 124 or 300).

[0265] It would be appreciated that the computing device 600 shown in Fig. 6 is merely
20 for purpose of illustration, without suggesting any limitation to the functions and scopes of the embodiments of the present disclosure in any manner.

[0266] As shown in Fig. 6, the computing device 600 includes a general-purpose computing device 600. The computing device 600 may at least comprise one or more processors or processing units 610, a memory 620, a storage unit 630, one or more
25 communication units 640, one or more input devices 650, and one or more output devices 660.

[0267] In some embodiments, the computing device 600 may be implemented as any user terminal or server terminal having the computing capability. The server terminal may be a server, a large-scale computing device or the like that is provided by a service
30 provider. The user terminal may for example be any type of mobile terminal, fixed terminal, or portable terminal, including a mobile phone, station, unit, device, multimedia

computer, multimedia tablet, Internet node, communicator, desktop computer, laptop computer, notebook computer, netbook computer, tablet computer, personal communication system (PCS) device, personal navigation device, personal digital assistant (PDA), audio/video player, digital camera/video camera, positioning device, television receiver, radio broadcast receiver, E-book device, gaming device, or any combination thereof, including the accessories and peripherals of these devices, or any combination thereof. It would be contemplated that the computing device 600 can support any type of interface to a user (such as “wearable” circuitry and the like).

[0268] The processing unit 610 may be a physical or virtual processor and can implement various processes based on programs stored in the memory 620. In a multi-processor system, multiple processing units execute computer executable instructions in parallel so as to improve the parallel processing capability of the computing device 600. The processing unit 610 may also be referred to as a central processing unit (CPU), a microprocessor, a controller or a microcontroller.

[0269] The computing device 600 typically includes various computer storage medium. Such medium can be any medium accessible by the computing device 600, including, but not limited to, volatile and non-volatile medium, or detachable and non-detachable medium. The memory 620 can be a volatile memory (for example, a register, cache, Random Access Memory (RAM)), a non-volatile memory (such as a Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), or a flash memory), or any combination thereof. The storage unit 630 may be any detachable or non-detachable medium and may include a machine-readable medium such as a memory, flash memory drive, magnetic disk or another other media, which can be used for storing information and/or data and can be accessed in the computing device 600.

[0270] The computing device 600 may further include additional detachable/non-detachable, volatile/non-volatile memory medium. Although not shown in Fig. 6, it is possible to provide a magnetic disk drive for reading from and/or writing into a detachable and non-volatile magnetic disk and an optical disk drive for reading from and/or writing into a detachable non-volatile optical disk. In such cases, each drive may be connected to a bus (not shown) via one or more data medium interfaces.

[0271] The communication unit 640 communicates with a further computing device via the communication medium. In addition, the functions of the components in the computing

device 600 can be implemented by a single computing cluster or multiple computing machines that can communicate via communication connections. Therefore, the computing device 600 can operate in a networked environment using a logical connection with one or more other servers, networked personal computers (PCs) or further general
5 network nodes.

[0272] The input device 650 may be one or more of a variety of input devices, such as a mouse, keyboard, tracking ball, voice-input device, and the like. The output device 660 may be one or more of a variety of output devices, such as a display, loudspeaker, printer, and the like. By means of the communication unit 640, the computing device 600 can
10 further communicate with one or more external devices (not shown) such as the storage devices and display device, with one or more devices enabling the user to interact with the computing device 600, or any devices (such as a network card, a modem and the like) enabling the computing device 600 to communicate with one or more other computing devices, if required. Such communication can be performed via input/output (I/O)
15 interfaces (not shown).

[0273] In some embodiments, instead of being integrated in a single device, some or all components of the computing device 600 may also be arranged in cloud computing architecture. In the cloud computing architecture, the components may be provided remotely and work together to implement the functionalities described in the present
20 disclosure. In some embodiments, cloud computing provides computing, software, data access and storage service, which will not require end users to be aware of the physical locations or configurations of the systems or hardware providing these services. In various embodiments, the cloud computing provides the services via a wide area network (such as Internet) using suitable protocols. For example, a cloud computing provider provides
25 applications over the wide area network, which can be accessed through a web browser or any other computing components. The software or components of the cloud computing architecture and corresponding data may be stored on a server at a remote position. The computing resources in the cloud computing environment may be merged or distributed at locations in a remote data center. Cloud computing infrastructures may provide the
30 services through a shared data center, though they behave as a single access point for the users. Therefore, the cloud computing architectures may be used to provide the components and functionalities described herein from a service provider at a remote location. Alternatively, they may be provided from a conventional server or installed

directly or otherwise on a client device.

[0274] The computing device 600 may be used to implement video encoding/decoding in embodiments of the present disclosure. The memory 620 may include one or more video coding modules 625 having one or more program instructions. These modules are
5 accessible and executable by the processing unit 610 to perform the functionalities of the various embodiments described herein.

[0275] In the example embodiments of performing video encoding, the input device 650 may receive video data as an input 670 to be encoded. The video data may be processed, for example, by the video coding module 625, to generate an encoded bitstream. The
10 encoded bitstream may be provided via the output device 660 as an output 680.

[0276] In the example embodiments of performing video decoding, the input device 650 may receive an encoded bitstream as the input 670. The encoded bitstream may be processed, for example, by the video coding module 625, to generate decoded video data. The decoded video data may be provided via the output device 660 as the output 680.

[0277] While this disclosure has been particularly shown and described with references
15 to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the present application as defined by the appended claims. Such variations are intended to be covered by the scope of this present application. As such, the foregoing
20 description of embodiments of the present application is not intended to be limiting.

I/We Claim:

1. A method for video processing, comprising:

performing a conversion between a video and a bitstream of the video, wherein a neural-
5 network post-processing filter (NNPF) is applied on at least one picture associated with the
video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first
candidate of a plurality of candidates for the purpose at least indicates that at least one of the
following is applicable: decreasing a width of the at least one picture, or decreasing a height of
the at least one picture.

10 2. The method of claim 1, wherein the first indication comprises a syntax element
nnpfc_purpose.

15 3. The method of any of claims 1-2, wherein the at least one picture comprises at least
one decoded picture or at least one cropped decoded picture of the video.

4. The method of any of claims 1-3, wherein the first candidate is resampling a
resolution of the at least one picture.

20 5. The method of claim 4, wherein resampling the resolution of the at least one picture
indicates that at least one of the following is applicable:

decreasing the width of the at least one picture,
decreasing the height of the at least one picture,
increasing the width of the at least one picture, or
25 increasing the height of the at least one picture.

6. The method of any of claims 1-5, wherein a scaling ratio of at least one of the width
or the height of the at least one picture is in a predetermined range.

30 7. The method of claim 6, wherein a lower limit of the predetermined range is equal to
1/16.

8. The method of any of claims 6-7, wherein an upper limit of the predetermined range
is equal to 16.

9. The method of claim 6 or 8, wherein a lower limit of the predetermined range is equal to one of the following: 1, 1/2, 1/4, or 1/8.

5 10. The method of any of claims 6-7 and 9, wherein an upper limit of the predetermined range is equal to one of the following: 1, 2, 4, or 8.

10 11. The method of any of claims 1-3, wherein the first candidate is downsampling a resolution of the at least one picture.

12. The method of claim 11, wherein the purpose of the NNPF is disallowed to comprise downsampling the resolution of the at least one picture and upsampling the resolution of the at least one picture.

15 13. The method of any of claims 11-12, wherein a downsampling ratio of at least one of the width or the height of the at least one picture is in a predetermined range.

20 14. The method of claim 13, wherein a lower limit of the predetermined range is equal to one of the following: 1, 1/2, 1/4, 1/8, or 1/16.

15. The method of any of claims 13-14, wherein an upper limit of the predetermined range is equal to 1.

25 16. The method of any of claims 1-15, wherein a second candidate of the plurality of candidates for the purpose at least indicates that at least one of the following is applicable:
decreasing a luma bit depth of the at least one picture, or
decreasing a chroma bit depth of the at least one picture.

30 17. The method of claim 16, wherein the second candidate is downsampling a bit depth of the at least one picture.

18. The method of claim 17, wherein the purpose of the NNPF is disallowed to comprise downsampling the bit depth of the at least one picture and upsampling the bit depth of the at least one picture.

19. The method of claim 16, wherein the second candidate is resampling a bit depth of the at least one picture.

5 20. The method of claim 19, wherein resampling the bit depth of the at least one picture indicates that at least one of the following is applicable:

 decreasing the luma bit depth of the at least one picture,
 decreasing the chroma bit depth of the at least one picture,
 increasing the luma bit depth of the at least one picture, or
10 increasing the chroma bit depth of the at least one picture.

21. The method of any of claims 1-20, wherein a third candidate of the plurality of candidates for the purpose at least indicates that downsampling a chroma format of the at least one picture is applicable.

15 22. The method of claim 21, wherein the third candidate is downsampling the chroma format of the at least one picture.

20 23. The method of claim 22, wherein the purpose of the NNPF is disallowed to comprise downsampling the chroma format of the at least one picture and upsampling the chroma format of the at least one picture.

24. The method of any of claims 22-23, wherein downsampling the chroma format of the at least one picture comprises at least one of the following:

25 changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format,
 changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format, or
 changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format.

25 25. The method of any of claims 22-23, wherein downsampling the chroma format of the at least one picture comprises at least one of the following:

 changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format,
 changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format, or
 changing the chroma format from 4:2:0 chroma format to 4:0:0 chroma format.

26. The method of claim 21, wherein the third candidate is resampling the chroma format of the at least one picture.

27. The method of claim 26, wherein resampling the chroma format of the at least one picture comprises at least one of the following:

changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format,
changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format,
changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format,
changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format,
changing the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or
changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format.

28. The method of claim 26, wherein resampling the chroma format of the at least one picture comprises at least one of the following:

changing the chroma format from 4:4:4 chroma format to 4:2:2 chroma format,
changing the chroma format from 4:4:4 chroma format to 4:2:0 chroma format,
changing the chroma format from 4:4:4 chroma format to 4:0:0 chroma format,
changing the chroma format from 4:2:2 chroma format to 4:2:0 chroma format,
changing the chroma format from 4:2:2 chroma format to 4:0:0 chroma format,
changing the chroma format from 4:0:0 chroma format to 4:2:0 chroma format,
changing the chroma format from 4:0:0 chroma format to 4:2:2 chroma format,
changing the chroma format from 4:0:0 chroma format to 4:4:4 chroma format,
changing the chroma format from 4:2:0 chroma format to 4:2:2 chroma format,
changing the chroma format from 4:2:0 chroma format to 4:4:4 chroma format, or
changing the chroma format from 4:2:2 chroma format to 4:4:4 chroma format.

29. The method of any of claims 1-28, wherein a fourth candidate of the plurality of candidates for the purpose at least indicates that downsampling a picture rate of the video is applicable.

30. The method of claim 29, wherein the fourth candidate is downsampling the picture rate of the video.

31. The method of claim 30, wherein the purpose of the NNPF is disallowed to comprise downsampling the picture rate of the video and upsampling the picture rate of the video.

5 32. The method of claim 29, wherein the fourth candidate is resampling the picture rate of the video.

33. The method of claim 32, wherein resampling the picture rate of the video indicates that downsampling the picture rate of the video or upsampling the picture rate of the video is applicable.

10 34. The method of any of claims 1-33, wherein the conversion includes encoding the video into the bitstream.

15 35. The method of any of claims 1-33, wherein the conversion includes decoding the video from the bitstream.

20 36. An apparatus for video processing comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method in accordance with any of claims 1-35.

37. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of claims 1-35.

25 38. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by an apparatus for video processing, wherein the method comprises:

performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture.

39. A method for storing a bitstream of a video, comprising:

performing a conversion between a video and a bitstream of the video, wherein a neural-network post-processing filter (NNPF) is applied on at least one picture associated with the video, the bitstream comprises a first indication indicating a purpose of the NNPF, and a first candidate of a plurality of candidates for the purpose at least indicates that at least one of the
5 following is applicable: decreasing a width of the at least one picture, or decreasing a height of the at least one picture; and

storing the bitstream in a non-transitory computer-readable recording medium.

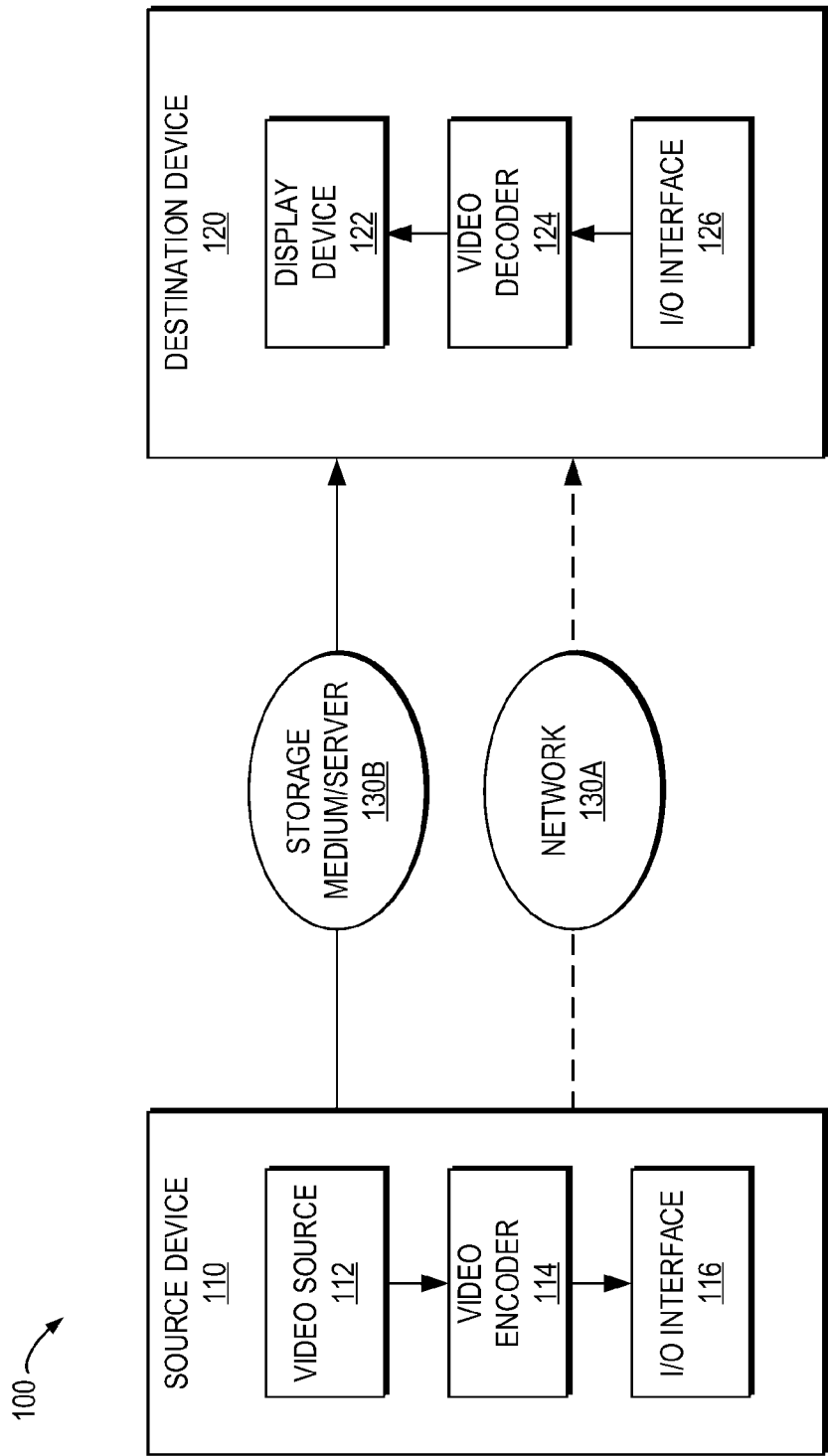


Fig. 1

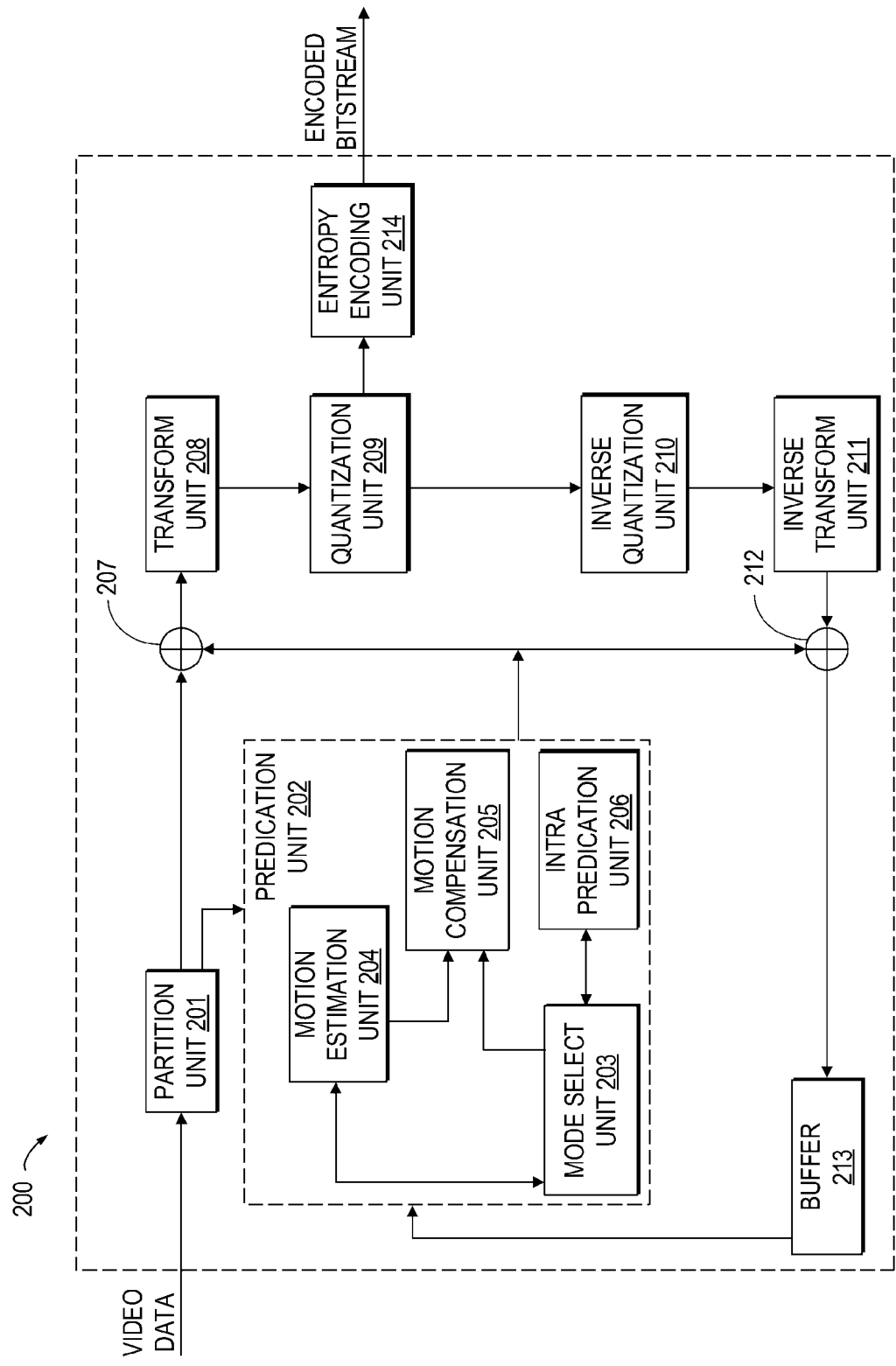


Fig. 2

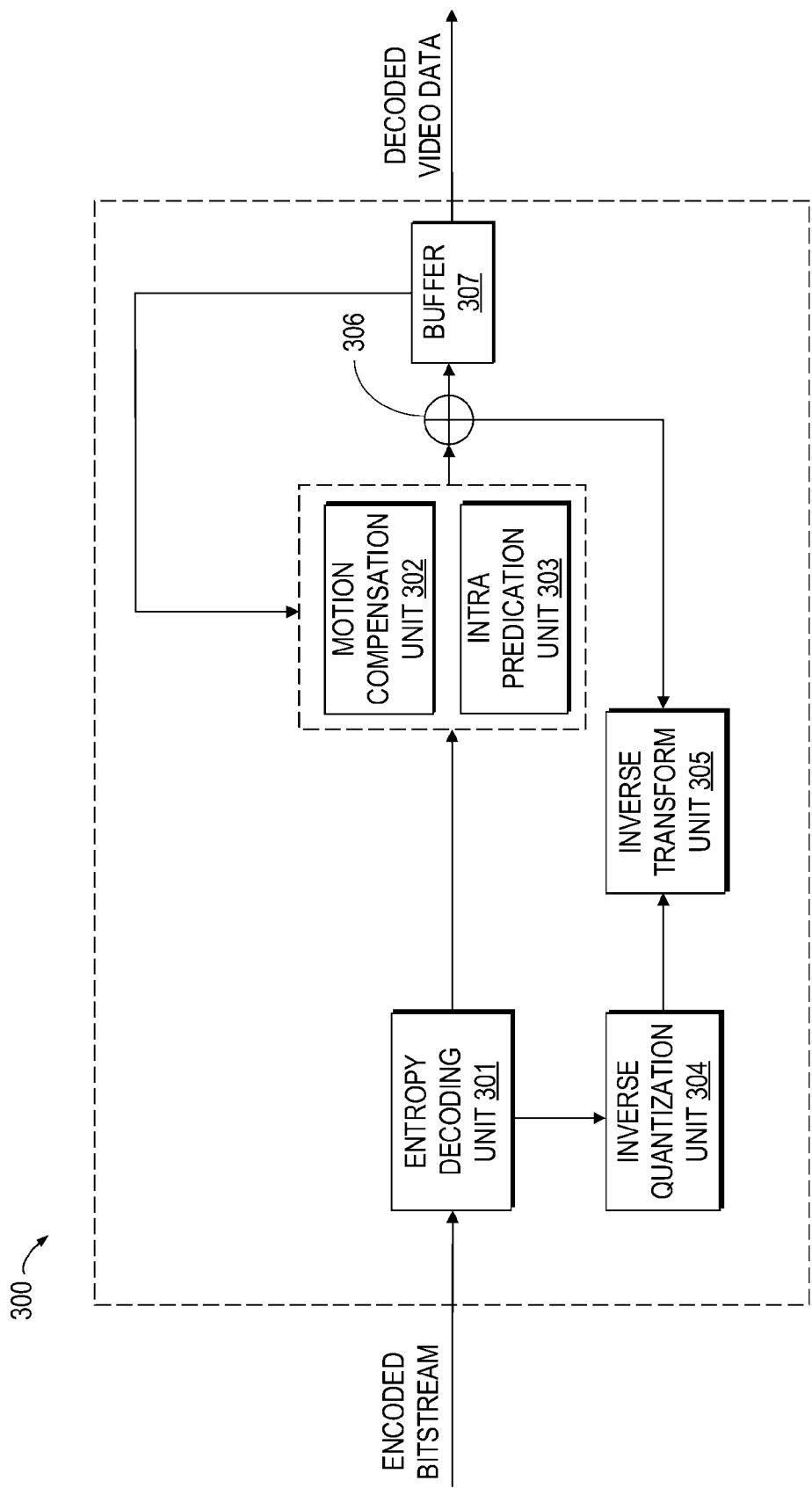


Fig. 3

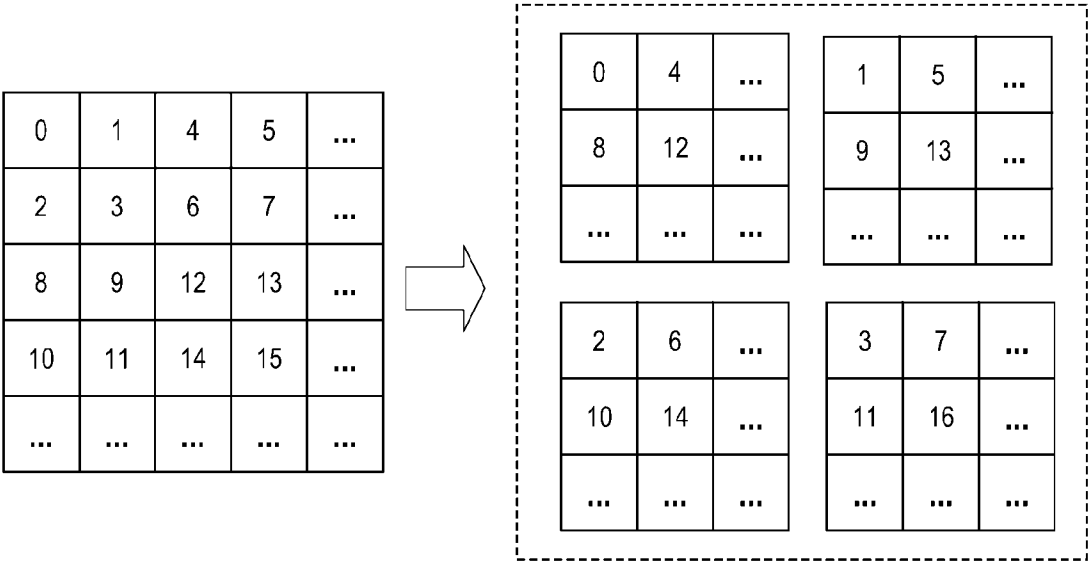


Fig. 4

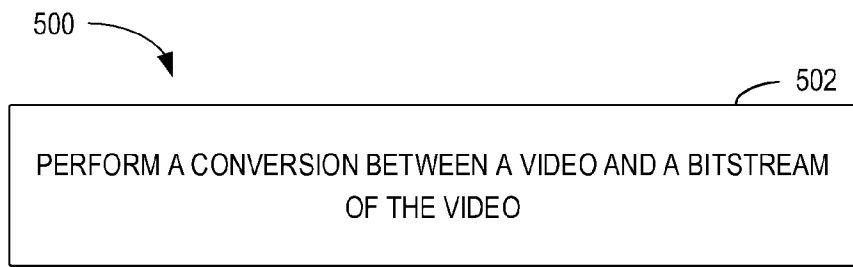


Fig. 5

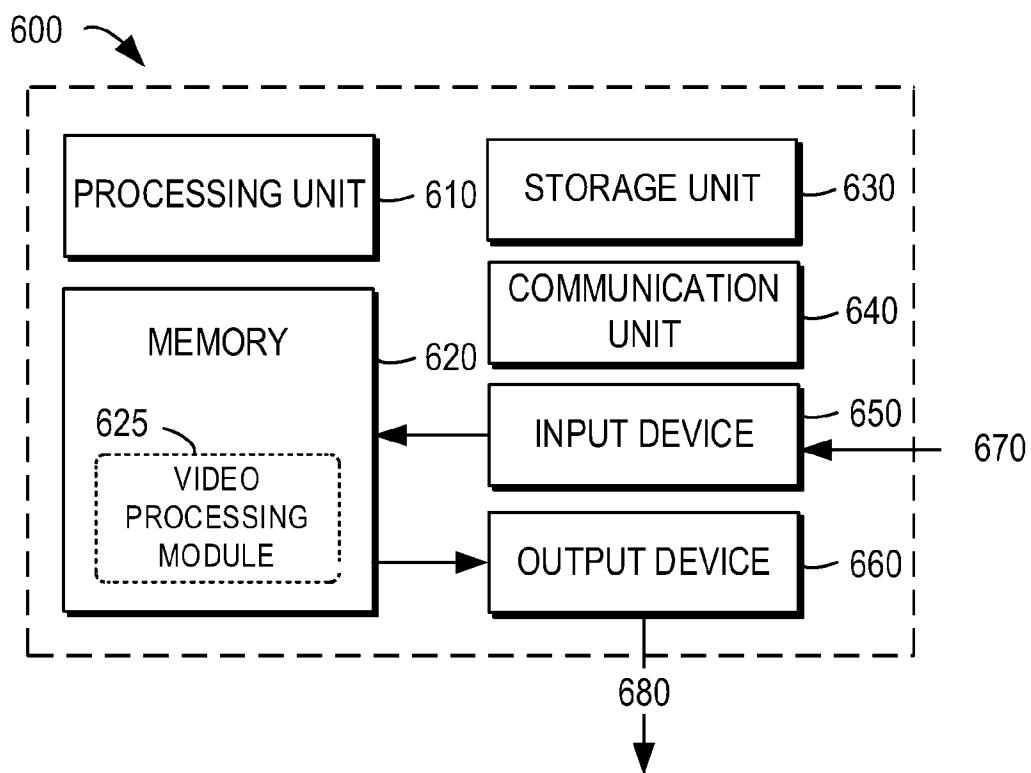


Fig. 6

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2024/086465

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/117(2014.01)i; H04N19/187(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC:H04N

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

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

CNTXT,ENTXT,ENTXTC,DWPI,IEEE,JVET:neural-network post-processing filter, NNPF, height, width, decreas+, downsampl+, purpose, SEI, size

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	WO 2024039680 A1 (BYTEDANCE INC.) 22 February 2024 (2024-02-22) description, paragraphs 0169-0171	1-39
Y	XU, Jizheng et al. "AHG9: Bit-masking based representation of nnpfc purpose" <i>Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29 29th Meeting, by teleconference, 11-20 January 2023, 05 January 2023 (2023-01-05),</i> Section 3	1-39
Y	WO 2022128105 A1 (HUAWEI TECHNOLOGIES CO., LTD. et al.) 23 June 2022 (2022-06-23) description, page 10, line 8-16; page 30, line 15-20	1-39
A	CN 111726637 A (TENCENT AMERICA LLC) 29 September 2020 (2020-09-29) the whole document	1-39
A	CN 115209143 A (LIANMENG CO., LTD.) 18 October 2022 (2022-10-18) the whole document	1-39
A	CN 115396666 A (LIANMENG CO., LTD.) 25 November 2022 (2022-11-25) the whole document	1-39

☒ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

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

"D" document cited by the applicant in the international application

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

06 June 2024

Date of mailing of the international search report

13 June 2024

Name and mailing address of the ISA/CN

CHINA NATIONAL INTELLECTUAL PROPERTY
ADMINISTRATION
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088, China

Authorized officer

CHENG, Cong

Telephone No. (+86) 010-53961725

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2024/086465

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 4161072 A1 (NOKIA TECHNOLOGIES OY) 05 April 2023 (2023-04-05) the whole document	1-39

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No. PCT/CN2024/086465

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
WO	2024039680	A1	22 February 2024	None	
WO	2022128105	A1	23 June 2022	TW 202228081 A	16 July 2022
				JP 2023547941 A	14 November 2023
				EP 4205390 A1	05 July 2023
				US 2023336784 A1	19 October 2023
				CN 116648912 A1	25 August 2023
CN	111726637	A	29 September 2020	US 2020304836 A1	24 September 2020
CN	115209143	A	18 October 2022	US 2022329837 A1	13 October 2022
CN	115396666	A	25 November 2022	US 2022394288 A1	08 December 2022
EP	4161072	A1	05 April 2023	US 2023112309 A1	13 April 2023