|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO CODING TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 175** |
| **Online – 11–20 January 2023** |
| |  |  | | --- | --- | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Title:** | **Improvements under consideration for neural network post-filter SEI messages** | | **Type:** | **Project** | | **Subtype:** | **Draft** | | **Status:** | **Approved** | | **Date:** | **2023-02-01** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **Pages:** | **27** (not including this cover page) | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **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 | Document: JVET-AC2032-v2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Improvements under consideration for neural network post filter SEI messages** | | |
| *Status:* | Output document approved by JVET | | |
| *Purpose:* | Draft text | | |
| *Author(s) or Contact(s):* | Sean McCarthy Sachin Deshpande Miska M. Hannuksela Hendry Gary J. Sullivan Ye-Kui Wang | Email: | sean.mccarthy@dolby.com sdeshpande@sharplabs.com miska.hannuksela@nokia.com dr.hendry@lge.com g.j.sullivan@outlook.com yekui.wang@bytedance.com |
| *Source:* | Editors | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document contains the draft text for changes to the versatile supplemental enhancement information messages for coded video bitstreams (VSEI) standard (Rec. ITU-T H.274 | ISO/IEC 23002-7), to specify additional SEI messages, including the shutter interval information SEI message, neural-network post-filter characteristics SEI message, neural-network post-filter activation SEI message, phase indication SEI message, and post-filter hint SEI message.

**Changes yet to be integrated:**

Items noted with adoptions in the BoG report JVET-AC0324-v4: none.

**Changes that have been integrated:**

Items noted with adoptions in the BoG report JVET-AC0324-v4: 1, 3, 4, 5, 6, 8, 11, 13, 14, 16, 17, 20, 21, 22, 23, 28, 29, 30, 31, 32, 33, 34.

**Changes to the specification text:**

*Clause 2*

Add the following references:

ISO/IEC 15938-17, Information technology – Multimedia content description interface – Part 17: Compression of neural networks for multimedia content description and analysis.

IETF Standard 66, Uniform Resource Identifiers (URI): Generic Syntax, <http://tools.ietf.org/html/rfc3986>

IETF RFC 4151, The 'tag' URI Scheme, October 2005, <http://tools.ietf.org/html/rfc4151>

*Clause 4*

Add the following abbreviated terms:

NNPF neural-network post-processing filter

NNPFA neural-network post-filter activation

NNPFC neural-network post-filter characteristics

URI uniform resource identifier

*Subclause 5.8*

Add the following function definitions in alphabetical order and adjust the intermediate and subsequent formula indices accordingly:

Reflect( x, y ) = (11)

Wrap( x, y ) = (17)

*Subclause 6.3*

Add the following function definition before the paragraph starting with "read\_bits( n )".

more\_data\_in\_payload( ) is specified as follows:

– If byte\_aligned( ) is equal to TRUE and the current position in an SEI message syntax structure or vui\_parameters( ) syntax structure is 8 \* payloadSize bits from the beginning of the syntax structure, the return value of more\_data\_in\_payload( ) is equal to FALSE.

– Otherwise, the return value of more\_data\_in\_payload( ) is equal to TRUE.

*Subclause 8.1*

Replace Table 4 with the following:

|  |  |
| --- | --- |
| **SEI message** | **Persistence scope** |
| Filler payload | The PU containing the SEI message |
| User data registered by Rec. ITU-T T.35 | Unspecified |
| User data unregistered | Unspecified |
| Film grain characteristics | Specified by the syntax of the SEI message |
| Frame packing arrangement | Specified by the syntax of the SEI message |
| Display orientation | Specified by the syntax of the SEI message |
| Referenced parameter sets | The CLVS containing the SEI message |
| Decoded picture hash | The PU containing the SEI message |
| Mastering display colour volume | The CLVS containing the SEI message |
| Colour transform information | Specified by the syntax of the SEI message |
| Content light level information | The CLVS containing the SEI message |
| DRAP indication | The picture associated with the SEI message |
| Alternative transfer characteristics | The CLVS containing the SEI message |
| Ambient viewing environment | The CLVS containing the SEI message |
| Content colour volume | Specified by the syntax of the SEI message |
| Equirectangular projection | Specified by the syntax of the SEI message |
| Generalized cubemap projection | Specified by the syntax of the SEI message |
| Sphere rotation | Specified by the syntax of the SEI message |
| Region-wise packing | Specified by the syntax of the SEI message |
| Omnidirectional viewport | Specified by the syntax of the SEI message |
| Alpha channel information | Specified by the syntax of the SEI message |
| Frame-field information | The PU containing the SEI message |
| Depth representation information | Specified by the semantics of the SEI message |
| Multiview acquisition information | The CVS containing the SEI message |
| Multiview view position | The CVS containing the SEI message |
| Annotated regions | Specified by the syntax of the SEI message |
| Sample aspect ratio information | Specified by the syntax of the SEI message |
| Scalability dimension information | The CVS containing the SEI message |
| Extended DRAP indication | The picture associated with the SEI message |
| Shutter interval information | The CVS containing the SEI message |
| Neural-network post-filter characteristics | The CVS containing the SEI message |
| Neural-network post-filter activation | Specified by the syntax of the SEI message |
| Phase indication | Specified by the semantics of the SEI message |
| Post-filter hint | Specified by the syntax of the SEI message |

*Subclause 8.24.2*

Replace the following paragraph:

The presence of the EDRAP indication SEI message indicates that the constraints on picture order and picture referencing specified in this subclause apply. These constraints can enable a decoder to properly decode the EDRAP picture and the pictures that are in the same layer and follow it in both decoding order and output order without needing to decode any other pictures in the same layer except the list of pictures referenceablePictures, which consists of a list of IRAP or EDRAP pictures in decoding order that are within the same CLVS and identified by the edrap\_ref\_rap\_id[ i ] syntax elements.

with the following:

The presence of the EDRAP indication SEI message indicates that the constraints on picture order and picture referencing specified in this subclause apply. These constraints can enable a decoder to properly decode the EDRAP picture and the pictures that are in the same layer and follow it in both decoding order and output order without needing to decode any other pictures in the same layer except the list of pictures referenceablePictures. The list referenceablePictures consists of the list of IRAP or EDRAP pictures that are present in the current CLVS and are identified by the edrap\_ref\_rap\_id[ i ] syntax elements, and these pictures are listed in decoder order in the list.

*New subclauses 8.27 to 8.31*

Add subclauses 8.27 to 8.31 as follows:

* 1. **Shutter interval information SEI message**

**8.27.1 Shutter interval information SEI message syntax**

|  |  |
| --- | --- |
| shutter\_interval\_info( payloadSize ) { | **Descriptor** |
| **sii\_time\_scale** | u(32) |
| **fixed\_shutter\_interval\_within\_clvs\_flag** | u(1) |
| if( fixed\_shutter\_interval\_within\_clvs\_flag ) |  |
| **sii\_num\_units\_in\_shutter\_interval** | u(32) |
| else { |  |
| **sii\_max\_sub\_layers\_minus1** | u(3) |
| for( i = 0; i <= sii\_max\_sub\_layers\_minus1; i++ ) |  |
| **sub\_layer\_num\_units\_in\_shutter\_interval**[ i ] | u(32) |
| } |  |
| } |  |

**8.27.2 Shutter interval information SEI message syntax**

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

**sii\_time\_scale** specifies the number of time units that pass in one second. The value of sii\_time\_scale shall not be equal to 0. For example, a time coordinate system that measures time using a 27 MHz clock has an sii\_time\_scale of 27 000 000.

**fixed\_shutter\_interval\_within\_clvs\_flag** equal to 1 specifies that the indicated shutter interval is the same for all temporal sublayers in the CLVS. fixed\_shutter\_interval\_within\_clvs\_flagequal to 0 specifies that the indicated shutter interval may not be the same for all temporal sublayers in the CLVS.

**sii\_num\_units\_in\_shutter\_interval**, when fixed\_shutter\_interval\_within\_clvs\_flag is equal to 1, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable shutterInterval, in units of seconds, is equal to the quotient of sii\_num\_units\_in\_shutter\_interval divided by sii\_time\_scale. For example, to represent a shutter interval equal to 0.04 seconds, sii\_time\_scale may be equal to 27 000 000 and sii\_num\_units\_in\_shutter\_interval may be equal to 1 080 000.

**sii\_max\_sub\_layers\_minus1** plus 1 specifies the maximum number of temporal sublayers that may be present in each CLVS referring to the SPS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2022-03 Annex D when sii\_max\_sub\_layers\_minus1 is equal to 1 and fixed\_shutter\_interval\_within\_clvs\_flag is equal to 0.

**sub\_layer\_num\_units\_in\_shutter\_interval**[ i ], when present, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS. The shutter interval for each picture with temporal sublayer identifier equal to i in the CLVS, denoted by the variable subLayerShutterInterval[ i ], in units of seconds, is equal to the quotient of sub\_layer\_num\_units\_in\_shutter\_interval[ i ] divided by sii\_time\_scale.

The variable subLayerShutterInterval[ i ], corresponding to the indicated shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS, is thus derived as follows:

if( fixed\_shutter\_interval\_within\_clvs\_flag )  
 subLayerShutterInterval[ i ] = sii\_num\_units\_in\_shutter\_interval ÷ sii\_time\_scale (75)  
else  
 subLayerShutterInterval[ i ] = sub\_layer\_num\_units\_in\_shutter\_interval[ i ] ÷ sii\_time\_scale

* 1. **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) |
| } |  |
| } |  |

* + 1. **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 BitDepthY for the luma sample array of the input pictures.

– Bit depth BitDepthC 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 NNPFdefined 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 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 NNPFC SEI message can be present for the same picture. When more than one NNPFC 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 NNPFC SEI messages with nnpfc\_purpose in the range of 64 to 65 535, inclusive.

**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 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) |
| ( nnpfc\_purpose & 0x02 )  !=  0 | With chroma upsampling |
| nnpfc\_purpose > 0  && ( nnpfc\_purpose & 0x04 )  = =  0 | 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 |

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 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 NNPFC 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.

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 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.

**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.

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, nnpfc\_mode\_idc equal to 1 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.

When an NNPFC SEI message is neither the first NNPFC SEI message, in decoding order, nor a repetition of the first NNPFC 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 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.

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 conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFC 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.

When this SEI message is the first NNPFC SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS, the NNPF PostProcessingFilter( ) is assigned to be the same as the base NNPF.

When this SEI message is neither the first NNPFC SEI message, in decoding order, nor a repetition of the irst NNPFC SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS, an NNPF PostProcessingFilter( ) is obtained by applying the update defined by this SEI message to the base NNPF.

Updates are not cumulative but rather each update is applied on the base NNPF, which is the NNPF specified by the first NNPFC SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS.

**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.

**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.

NOTE 3 – nnpfc\_tag\_uri enables uniquely identifying the format of neural network data specified by nnrpf\_uri without needing a central registration authority.

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.

**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.

**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.

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.

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.

**nnpfc\_base\_flag** equal to 1 specifies that the SEI message specifies the base NNPF. nnpf\_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.

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 paylaod content of nnpfcB shall be the same as that of nnpfcA.

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.

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.

**nnpfc\_out\_sub\_c\_flag** specifies the values of the variables outSubWidthC and outSubHeightC when nnpfc\_purpose & 0x02 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 and nnpfc\_out\_sub\_c\_flag is present, the value of nnpfc\_out\_sub\_c\_flag shall be equal to 1.

**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.

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 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.

**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.

**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.

**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.

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++ ) (76)  
 numOutputPics  +=  nnpfc\_interpolated\_pics[ i ]  
} else  
 numOutputPics = 1

**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.

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.

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 current channel during the process.

**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:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) − 1 ) (77)

InpC( x )= x ÷ ( ( 1  <<  BitDepthC ) − 1 ) (78)

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:

shiftY = BitDepthY − inpTensorBitDepthY  
if( inpTensorBitDepthY >= BitDepthY)  
 InpY( x ) = x  <<  ( inpTensorBitDepthY − BitDepthY ) (79)  
else  
 InpY( x ) = Clip3(0, ( 1  <<  inpTensorBitDepthY ) − 1, ( x + ( 1  <<  ( shiftY − 1 ) ) )  >>  shiftY )

shiftC = BitDepthC − inpTensorBitDepthC  
if( inpTensorBitDepthC >= BitDepthC )  
 InpC( x ) = x  <<  ( inpTensorBitDepthC − BitDepthC ) (80)  
else  
 InpC( x ) = Clip3(0, ( 1  <<  inpTensorBitDepthC ) − 1, ( x + ( 1  <<  ( shiftC − 1 ) ) )  >>  shiftC )

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.

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 NNPFC SEI messages that contain reserved values of nnpfc\_inp\_format\_idc.

**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:

inpTensorBitDepthY = nnpfc\_inp\_tensor\_luma\_bitdepth\_minus8 + 8(81)

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.

**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:

inpTensorBitDepthC = nnpfc\_inp\_tensor\_chroma\_bitdepth\_minus8 + 8(82)

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.

**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.

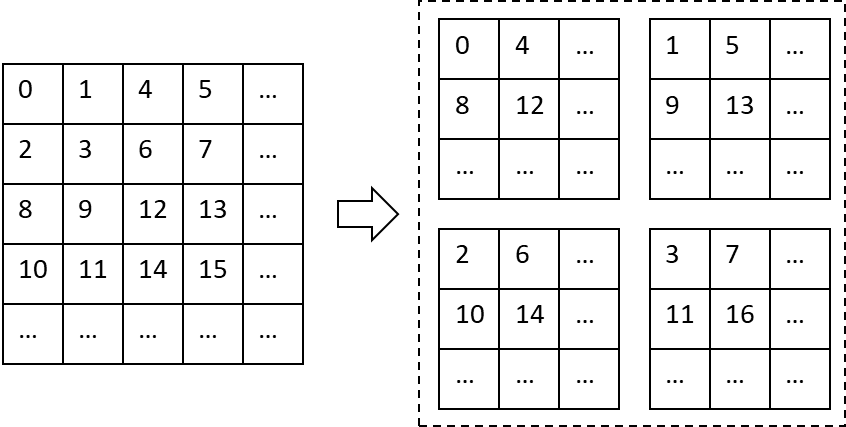
The value of nnpfc\_inp\_order\_idc shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this edition of this document. Values of 4 to 255, inclusive, for nnpfc\_inp\_order\_idc are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFC 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.

When ChromaFormatIdc is not equal to 1, nnpfc\_inp\_order\_idc shall not be equal to 3.

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 Figure 12. This nnpfc\_inp\_order\_idc can only be used when the input chroma format is 4:2:0. |
| 4..255 | Reserved |



**Figure 12 – Illustration of deriving the four luma channles (right) from the luma component (left) when nnpfc\_inp\_order\_idc is equal to 3**

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

**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.

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 NNPFC 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.

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

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  
 inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 1 ] = strengthControlScaledVal  
 }  
 else if( nnpfc\_inp\_order\_idc = = 1 ) (84)  
 for( yP = −nnpfc\_overlap; yP < inpPatchHeight + nnpfc\_overlap; yP++)  
 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 / SubHeightC,  
 CroppedWidth / SubWidthC, CroppedCrPic[ i ] ) )  
 yPovlp = yP + nnpfc\_overlap  
 xPovlp = xP + nnpfc\_overlap  
 if( !nnpfc\_component\_last\_flag ) {  
 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  
 }  
 if( nnpfc\_auxiliary\_inp\_idc = = 1 )  
 if( !nnpfc\_component\_last\_flag )  
 inputTensor[ 0 ][ i ][ 2 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal  
 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  
 yC = yY / SubHeightC  
 xC = xY / SubWidthC  
 inpYVal = InpY( InpSampleVal( yY, xY, CroppedHeight,  
 CroppedWidth, CroppedYPic[ i ] ) )  
 inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,  
 CroppedWidth / SubWidthC, CroppedCbPic[ i ] ) )  
 inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,  
 CroppedWidth / SubWidthC, CroppedCrPic[ i ] ) )  
 yPovlp = yP + nnpfc\_overlap  
 xPovlp = xP + nnpfc\_overlap  
 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 {  
 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 )  
 if( !nnpfc\_component\_last\_flag )  
 inputTensor[ 0 ][ i ][ 3 ][ yPovlp ][ xPovlp ] = strengthControlScaledVal  
 else  
 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  
 yC = cTop / 2 + yP  
 xC = cLeft / 2 + xP  
 inpTLVal = InpY( InpSampleVal( yTL, xTL, CroppedHeight,  
 CroppedWidth, CroppedYPic[ i ] ) )  
 inpTRVal = InpY( InpSampleVal( yTL, xBR, CroppedHeight,  
 CroppedWidth, CroppedYPic[ i ] ) )  
 inpBLVal = InpY( InpSampleVal( yBR, xTL, CroppedHeight,  
 CroppedWidth, CroppedYPic[ i ] ) )  
 inpBRVal = InpY( InpSampleVal( yBR, xBR, CroppedHeight,  
 CroppedWidth, CroppedYPic[ i ] ) )  
 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  
 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  
 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  
 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 ] = strengthControlScaledVal  
 else  
 inputTensor[ 0 ][ i ][ yPovlp ][ xPovlp ][ 6 ] = strengthControlScaledVal  
 }  
}

**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 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.

**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 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 NNPFC SEI message, the value of nnpfc\_colour\_primaries is inferred to be equal to vui\_colour\_primaries.

**nnpfc\_transfer\_characteristics** has the same semantics as specified in subclause 7.3 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 NNPFC SEI message, the value of nnpfc\_transfer\_characteristics is inferred to be equal to vui\_transfer\_characteristics.

**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.

**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  <<  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 are unsigned integer numbers in the range of 0 to ( 1  <<  ( 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  <<  ( 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 NNPFC 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.

**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.

– nnpfc\_out\_tensor\_chroma\_bitdepth\_minus8 + 8 is greater than BitDepthC.

**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 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.

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 )  
 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 ] }  
 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  
 if ( ySrc < nnpfc\_pic\_height\_in\_luma\_samples / outSubHeightC &&  
 xSrc < nnpfc\_pic\_width\_in\_luma\_samples / outSubWidthC )  
 if( !nnpfc\_component\_last\_flag ) {  
 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 ]  
 }  
 }  
 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  
 yC = yY / outSubHeightC   
 xC = xY / outSubWidthC   
 yPc = ( yP / outSubHeightC ) \* outSubHeightC  
 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 ]  
 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 )  
 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 &&  
 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 ] = 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 ]  
 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 ]  
 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 ]  
 }  
 }  
}

**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. 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 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.

**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.

**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 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 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.

– The value of inpPatchWidth + 2 \* nnpfc\_overlap shall be a positive integer multiple ofnnpfc\_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 ofnnpfc\_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:

outPatchWidth = ( nnpfc\_pic\_width\_in\_luma\_samples \* inpPatchWidth ) / CroppedWidth (86)

outPatchHeight = ( nnpfc\_pic\_height\_in\_luma\_samples \* inpPatchHeight ) / CroppedHeight (87)

horCScaling = SubWidthC / outSubWidthC (88)

verCScaling = SubHeightC / outSubHeightC (89)

outPatchCWidth = outPatchWidth \* horCScaling (90)

outPatchCHeight = outPatchHeight \* verCScaling (91)

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.

**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.

**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 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 inference engines.

if( nnpfc\_padding\_type = = 0 )  
 if( y < 0 | | x < 0 | | y >= picHeight | | x >= picWidth )  
 sampleVal = 0  
 else  
 sampleVal = croppedPic[ x ][ y ] (92)  
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 )   
 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  
 sampleVal[ 2 ] = nnpfc\_cr\_padding\_val  
 else  
 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:

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 )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 1 )  
 for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop += inpPatchHeight )  
 for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft += inpPatchWidth ) { (93)  
 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 )  
 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 NNPFC 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 NNPFC 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:

maxNumParameters = ( 2 048  <<  nnpfc\_num\_parameters\_idc ) − 1(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 unknown. The value of nnpfc\_num\_kmac\_operations\_idc shall be in the range of 0 to 232 − 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 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.

**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.

**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.

* 1. **Neural-network post-filter activation SEI message**
     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) |
| } |  |

* + 1. **Neural-network post-filter activation SEI message semantics**

The neural-network post-filter activation (NNPFA) SEI message activates or de-activates 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.

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.

**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.

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.

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.

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.

**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.

**nnpfa\_persistence\_flag** specifies the persistence of the target NNPF for the current layer.

nnpfa\_persistence\_flag equal to 0 specifies that the target NNPF may be used for post-processing filtering for the current picture only.

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.

– The bitstream ends.

– A picture in the current layer associated with a NNPFA 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.

NOTE 2 – The target NNPF is not applied for this subsequent picture in the current layer associated with a NNPFA SEI message with the same nnpfa\_target\_id as the current SEI message and nnpfa\_cancel\_flag equal to 1.

Let the nnpfcTargetPictures be the set of pictures to which the last NNPFC SEI message with nnpfc\_id equal to nnpfa\_target\_id that precedes the current NNPFA SEI message in decoding order pertains. Let nnpfaTargetPictures be the set of pictures for which the target NNPF is activated by the current NNPFA SEI message. It is a requirement of bitstream conformance that any picture included in nnpfaTargetPictures shall also be included in nnpfcTargetPictures.

* 1. **Phase indication SEI message**
     1. **Phase indication SEI message syntax**

|  |  |
| --- | --- |
| phase\_indication( payloadSize ) { | **Descriptor** |
| **hor\_phase\_num** | u(8) |
| **hor\_phase\_den\_minus1** | u(8) |
| **ver\_phase\_num** | u(8) |
| **ver\_phase\_den\_minus1** | u(8) |
| } |  |

* + 1. **Phase indication SEI message semantics**

The phase indication SEI message provides the decoder with information about the position of luma sampling locations in cropped decoded pictures relative to a rendering window. This information may be used by a decoder to ensure the correct spatial alignment of rendered pictures, for example when switching between picture resolutions.



**Figure 13: The ratios and represent the horizontal and vertical locations of the luma samples (marked with x) relative to a rendering window. is equal to hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ), and to ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ).**

NOTE 1 – When the number of luma output samples in horizontal direction is equal to the width of the rendering window, and hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ) is equal to 1÷2, the picture is intended to be rendered without applying any horizontal phase shift. Correspondingly, when the number of luma output samples in vertical direction is equal to the height of the rendering window, and ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ) is equal to 1÷2, the picture is intended to be rendered without applying any vertical phase shift.

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

– A cropped picture width and picture height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

The phase indication SEI message applies to the current cropped decoded picture and persists for all subsequent pictures of the current layer in output order with the same value of CroppedWidth as the current picture and the same value of CroppedHeight as the current picture until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer with an associated phase indication SEI message and the same value of CroppedWidth as the current picture and the same value of CroppedHeight as the current picture is output and follows the current picture in output order.

**hor\_phase\_num** and **hor\_phase\_den\_minus1** specify the horizontal position of luma sampling locations relative to a rendering window. The horizontal position hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ) is expressed in units of the horizontal distance between two horizontally adjacent luma sampling locations. The value of hor\_phase\_num shall be greater than or equal to 0 and less than or equal to hor\_phase\_den\_minus1 + 1.

**ver\_phase\_num** and **ver\_phase\_den\_minus1** specify the vertical position of luma sampling locations relative to a rendering window. The vertical position ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ) is expressed in units of the vertical distance between two vertically adjacent luma sampling locations. The value of ver\_phase\_num shall be greater than or equal to 0 and less than or equal to ver\_phase\_den\_minus1 + 1.

NOTE 2 – The phase indicators can be used during the rendering process. For example, texture coordinates can be offset by an amount proportional to the signalled horizontal and vertical phase indicators.

NOTE 3 – The signalled phase indicators applies to the luma samples of the decoded pictures. The phase offset for chroma samples can be derived from the signalled phase indicators taking into account the chroma sample location relative to luma sample location as indicated by ChromaFormatIdc, vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field. When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) and the value of vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field, as applicable, are equal to 6 or are inferred to be equal to 6, the nominal vertical and horizontal relative locations of luma and chroma samples that corresponds to vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field equal to 0 may be assumed.

* 1. **Post-filter hint SEI message**
     1. **Post-filter hint SEI message syntax**

|  |  |
| --- | --- |
| post\_filter\_hint( payloadSize ) { | **Descriptor** |
| **filter\_hint\_cancel\_flag** | u(1) |
| if( !filter\_hint\_cancel\_flag ) { |  |
| **filter\_hint\_persistence\_flag** | u(1) |
| **filter\_hint\_size\_y** | ue(v) |
| **filter\_hint\_size\_x** | ue(v) |
| **filter\_hint\_type** | u(2) |
| **filter\_hint\_chroma\_coeff\_present\_flag** | u(1) |
| for( cIdx = 0; cIdx < ( filter\_hint\_chroma\_coeff\_present\_flag ? 3 : 1 ); cIdx++ ) |  |
| for( cy = 0; cy < filter\_hint\_size\_y; cy++ ) |  |
| for( cx = 0; cx < filter\_hint\_size\_x; cx++ ) |  |
| **filter\_hint\_value**[ cIdx ][ cy ][ cx ] | se(v) |
| } |  |
| } |  |

* + 1. **Post-filter hint SEI message semantics**

This SEI message provides the coefficients of a post-filter or correlation information for the design of a post-filter for potential use in post-processing of a set of pictures after they are decoded and output to obtain improved displayed quality.

Use of the post-filter hint SEI message requires the definition of the following variables:

– Cropped decoded output picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– Bit depth BitDepthY for the luma sample array of the cropped decoded output picture.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the cropped decoded output picture.

– Sample arrays CroppedSampleArray[ cIdx ] for the cIdx-th component of the cropped decoded output picture.

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

**filter\_hint\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous post-filter hint SEI message in output order that applies to the current layer. filter\_hint\_cancel\_flag equal to 0 indicates that post-filter hint information follows.

**filter\_hint\_persistence\_flag** specifies the persistence of the post-filter hint SEI message for the current layer. filter\_hint\_persistence\_flag equal to 0 specifies that the post-filter hint applies to the current decoded picture only. filter\_hint\_persistence\_flag equal to 1 specifies that the post-filter hint SEI message applies to the current decoded picture and persists for 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.
* The bitstream ends.
* A picture in the current layer in an AU associated with a post-filter hint SEI message is output that follows the current picture in output order.

**filter\_hint\_size\_y** specifies the vertical size of the filter coefficient or correlation array. The value of filter\_hint\_size\_y shall be in the range of 1 to 15, inclusive.

**filter\_hint\_size\_x** specifies the horizontal size of the filter coefficient or correlation array. The value of filter\_hint\_size\_x shall be in the range of 1 to 15, inclusive.

**filter\_hint\_type** identifies the type of the transmitted filter hints as specified in Table 24. The value of filter\_hint\_type shall be in the range of 0 to 2, inclusive. The value of filter\_hint\_type 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 post-filter hint SEI messages having filter\_hint\_type equal to 3.

**Table 24 – filter\_hint\_type values**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Coefficients of a 2D-FIR filter |
| 1 | Coefficients of two 1D-FIR filters |
| 2 | Cross-correlation matrix |

**filter\_hint\_chroma\_coeff\_present\_flag** equal 1 specifies that filter coefficients for chroma are present. filter\_hint\_chroma\_coeff\_present\_flag equal 0 specifies that filter coefficients for chroma are not present.

**filter\_hint\_value**[ cIdx ][ cy ][ cx ] specifies a filter coefficient or an element of a cross-correlation matrix between the original and the decoded signal with 16-bit precision. The value of filter\_hint\_value[ cIdx ][ cy ][ cx ] shall be in the range of −231 + 1 to 231 − 1, inclusive. cIdx specifies the related colour component, cy represents a counter in vertical direction and cx represents a counter in horizontal direction. Depending on the value of filter\_hint\_type, the following applies:

– If filter\_hint\_type is equal to 0, the coefficients of a 2-dimensional finite impulse response (FIR) filter with the size of filter\_hint\_size\_y \* filter\_hint\_size\_x are transmitted.

– Otherwise, if filter\_hint\_type is equal to 1, the filter coefficients of two 1-dimensional FIR filters are transmitted. In this case, filter\_hint\_size\_y shall be equal to 2. The index cy equal to 0 specifies the filter coefficients of the horizontal filter and cy equal to 1 specifies the filter coefficients of the vertical filter. In the filtering process, the horizontal filter is applied first and the result is filtered by the vertical filter.

– Otherwise (filter\_hint\_type is equal to 2), the transmitted hints specify a cross-correlation matrix between the original signal s and the decoded signal s′.

NOTE 1 – The normalized cross-correlation matrix for a related colour component identified by cIdx with the size of filter\_hint\_size\_y \* filter\_hint\_size\_x is defined as follows:

 (95)

where s denotes array of samples of the colour component cIdx of the original picture, s′ denotes CroppedSampleArray[ cIdx ], h is equal to CroppedHeight when cIdx is equal to 0 and equal to CroppedHeight / SubHeightC when cIdx is greater than 0, wis equal to CroppedWidth when cIdx is equal to 0 and equal to CroppedWidth / SubWidthC when cIdx is greater than 0, bitDepth is equal to BitDepthY when cIdx is equal to 0 and equal to BitDepthC when cIdx is greater than 0, OffsetY is equal to ( filter\_hint\_size\_y  >>  1 ), OffsetX is equal to ( filter\_hint\_size\_x  >>  1 ), 0  <=  cy < filter\_hint\_size\_y and 0  <=  cx < filter\_hint\_size\_x.

NOTE 2 – A decoder can derive a Wiener post-filter from the cross-correlation matrix of original and decoded signal and the auto-correlation matrix of the decoded signal.

*Bibliography*

Add a bibliographic reference as follows:

[16] ATSC A/341:2022-03, *ATSC Standard: Video – HEVC*.