

LOW COMPLEXITY NN-BASED IN LOOP FILTER ARCHITECTURES WITH SEPARABLE CONVOLUTION

[0001] This application claims priority to U.S. Patent Application No. 18/738,842, filed June 10, 2024, and U.S. Provisional Patent Application No. 63/507,651, filed June 12, 2023, the entire content of each of which are incorporated by reference. U.S. Patent Application No. 18/738,842, filed June 10, 2024, claims the benefit of U.S. Provisional Patent Application No. 63/507,651, filed June 12, 2023.

TECHNICAL FIELD

[0002] This disclosure relates to video encoding and video decoding.

BACKGROUND

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

[0004] Video coding techniques include spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (e.g., a video picture or a portion of a video picture) may be partitioned into video blocks, which may also be referred to as coding tree units (CTUs), coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or

B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

SUMMARY

[0005] In general, this disclosure describes techniques for neural network (NN)-based in-loop filtering for video coding. More particularly, this disclosure describes examples of a simplified NN-based in-loop filter. A Unified Architecture (UA) for NN-based in-loop filters has been developed. However, the UA is power and processing intensive and is therefore not appropriate for use on power and/or processing constrained devices, such as battery powered mobile devices. The techniques of this disclosure provide for a NN-based in-loop filter operable on such a constrained device.

[0006] In one example, a method includes: receiving a picture of the video data; reconstructing the picture of the video data; and applying a neural network (NN)-based filter to the reconstructed picture of the video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0007] In another example, a device includes one or more memories configured to store the video data; and one or more processors, the one or more processors configured to: receive a picture of video data; reconstruct the picture of video data; and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0008] In another example, a device includes means for receiving a picture of video data; means for reconstructing the picture of video data; and means for applying a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the

transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0009] In another example, computer-readable storage media is encoded with instructions that, when executed, cause one or more programmable processors to: receive a picture of video data; reconstruct the picture of video data; and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

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

BRIEF DESCRIPTION OF DRAWINGS

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

[0012] FIG. 2 is a block diagram illustrating an example hybrid video coding framework.

[0013] FIG. 3 illustrates an example hierarchical prediction structure with a group of pictures (GOP) size equal to 16.

[0014] FIG. 4 is a block diagram illustrating an example convolution neural network (CNN)-based filter with four layers.

[0015] FIG. 5 is a block diagram illustrating an example CNN-based filter with padded input samples and supplementary data.

[0016] FIG. 6 is a block diagram illustrating another example CNN-based filter with padded input samples and supplementary data.

[0017] FIG. 7 is a conceptual diagram illustrating an attention residual block of FIG. 6.

[0018] FIG. 8 is a block diagram illustrating another example CNN-based filter with padded input samples and supplementary data.

[0019] FIG. 9 is a block diagram illustrating an example of a simplified CNN-based filter architecture with padded input samples and supplementary data.

[0020] FIG. 10 is a conceptual diagram illustrating an example residual block structure of FIG. 9.

[0021] FIG. 11 is a block diagram illustrating another example of a simplified CNN-based filter architecture with padded input samples and supplementary data.

[0022] FIG. 12 is a conceptual diagram illustrating an example filter block structure of FIG. 11.

[0023] FIG. 13 is a block diagram illustrating an example CNN decomposition architecture.

[0024] FIG. 14 is a block diagram illustrating an example multiscale feature extraction backbone network with two-component convolution.

[0025] FIG. 15 is a block diagram illustrating an example unified filter with a joined model (joined luma and chroma).

[0026] FIG. 16 is a block diagram illustrating an example backbone block.

[0027] FIG. 17 is a block diagram illustrating an example unified filter with separate luma and chroma models (luma).

[0028] FIG. 18 is a block diagram illustrating an example unified filter with separate luma and chroma models (chroma).

[0029] FIG. 19 is a block diagram illustrating an example low complexity in-loop filter network architecture according to one or more aspects of this disclosure.

[0030] FIG. 20 is a block diagram illustrating an example backbone block according to one or more aspects of this disclosure.

[0031] FIG. 21 is a block diagram illustrating an example CP decomposition according to one or more aspects of this disclosure.

[0032] FIG. 22 is a flow diagram illustrating example NN-based in-loop filtering techniques according to one or more aspects of this disclosure.

[0033] FIG. 23 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0034] FIG. 24 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0035] FIG. 25 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure.

[0036] FIG. 26 is a flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure.

DETAILED DESCRIPTION

[0037] Video coding (e.g., video encoding and/or video decoding) typically involves predicting a block of video data from either an already coded block of video data in the same picture (e.g., intra prediction) or an already coded block of video data in a different picture (e.g., inter prediction). In some instances, the video encoder also calculates residual data by comparing the prediction block to the original block. Thus, the residual data represents a difference between the prediction block and the original block. To reduce the number of bits needed to signal the residual data, the video encoder transforms and quantizes the residual data and signals the transformed and quantized residual data in the encoded bitstream. The compression achieved by the transform and quantization processes may be lossy, meaning that transform and quantization processes may introduce distortion into the decoded video data.

[0038] A video decoder decodes and adds the residual data to the prediction block to produce a reconstructed video block that matches the original video block more closely than the prediction block alone. Due to the loss introduced by the transforming and quantizing of the residual data, the reconstructed block may have distortion or artifacts. One common type of artifact or distortion is referred to as blockiness, where the boundaries of the blocks used to code the video data are visible.

[0039] To further improve the quality of decoded video, a video decoder can perform one or more filtering operations on the reconstructed video blocks. Examples of these filtering operations include deblocking filtering, sample adaptive offset (SAO) filtering, and adaptive loop filtering (ALF). Parameters for these filtering operations may either be determined by a video encoder and explicitly signaled in the encoded video bitstream or may be implicitly determined by a video decoder without needing the parameters to be explicitly signaled in the encoded video bitstream.

[0040] This disclosure describes neural network (NN)-based in-loop filtering (ILF) architectures for purposes of video coding. The techniques of this disclosure may improve coding performance under complexity and memory requirements constraints. In some examples, the architectures of this disclosure may be convolution neural network (CNN)-based. In some CNN ILFs, complexity reduction may be achieved by decomposition (e.g., utilizing separable convolutions) in the place of conventional multi-dimensional convolutions.

[0041] In general, this disclosure describes techniques for NN-based in-loop filtering for video coding. Video coders (e.g., encoders and/or decoders) may have complexity and memory requirement constraints, such as when such video coders are implemented in a mobile device. The described techniques may improve coding performance under complexity and memory requirement constraints, thereby resulting in higher quality video playback than existing techniques under such constraints. Complexity reduction for the architecture may be achieved by utilizing a Canonical Polyadic (CP) decomposition with separable convolution in place of a conventional convolution with multidimensional kernels, removing or simplifying activation layers, and/or removing or moving multiscale branch(es). The techniques described in this disclosure are related to NN-based in-loop filtering, however, they are applicable to any NN-based video coding tool that consumes input data with certain statistical properties. The techniques may be used in the context of advanced video codecs, such as extensions of VVC or the next generation of video coding standards, and any other video codecs.

[0042] FIG. 1 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, unencoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0043] As shown in FIG. 1, system 100 includes a source device 102 that provides encoded video data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the video data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116 may be or include any of a wide range of devices, such as desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0044] In the example of FIG. 1, source device 102 includes video source 104, memory 106, video encoder 200, and output interface 108. Destination device 116 includes input interface 122, video decoder 300, memory 120, and display device 118. In accordance

with this disclosure, video encoder 200 of source device 102 and video decoder 300 of destination device 116 may be configured to apply the techniques for NN-based in-loop filtering. Thus, source device 102 represents an example of a video encoding device, while destination device 116 represents an example of a video decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 102 may receive video data from an external video source, such as an external camera. Likewise, destination device 116 may interface with an external display device, rather than include an integrated display device.

[0045] System 100 as shown in FIG. 1 is merely one example. In general, any digital video encoding and/or decoding device may perform techniques for NN-based in-loop filtering. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates coded video data for transmission to destination device 116. This disclosure refers to a “coding” device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder 200 and video decoder 300 represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, source device 102 and destination device 116 may operate in a substantially symmetrical manner such that each of source device 102 and destination device 116 includes video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between source device 102 and destination device 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0046] In general, video source 104 represents a source of video data (i.e., raw, unencoded video data) and provides a sequential series of pictures (also referred to as “frames”) of the video data to video encoder 200, which encodes data for the pictures. Video source 104 of source device 102 may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 104 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In each case, video encoder 200 encodes the captured, pre-captured, or computer-generated video data. Video encoder 200 may rearrange the pictures from the received order (sometimes referred to as “display order”) into a coding order for coding. Video encoder 200 may generate a bitstream including encoded video data. Source device 102 may then output the encoded video data

via output interface 108 onto computer-readable medium 110 for reception and/or retrieval by, e.g., input interface 122 of destination device 116.

[0047] Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some examples, memories 106, 120 may store raw video data, e.g., raw video from video source 104 and raw, decoded video data from video decoder 300. Additionally, or alternatively, memories 106, 120 may store software instructions executable by, e.g., video encoder 200 and video decoder 300, respectively. Although memory 106 and memory 120 are shown separately from video encoder 200 and video decoder 300 in this example, it should be understood that video encoder 200 and video decoder 300 may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded video data, e.g., output from video encoder 200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

[0048] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may include any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0049] In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 may include any of a variety of distributed or locally accessed data storage media such as a

hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data.

[0050] In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download.

[0051] File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast Multicast Service (MBMS) or Enhanced MBMS (eMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

[0052] Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

[0053] Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 include wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 includes a wireless transmitter, output interface 108 and input interface 122 may be configured to

transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

[0054] The techniques of this disclosure may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications.

[0055] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded video bitstream may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0056] Although not shown in FIG. 1, in some examples, video encoder 200 and video decoder 300 may each be integrated with an audio encoder and/or audio decoder (e.g., audio codec), and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and video in a common data stream. Example audio codecs may include AAC, AC-3, AC-4, ALAC, ALS, AMBE, AMR, AMR-WB (G.722.2), AMR-WB+, aptx (various versions), ATRAC, BroadVoice (BV16, BV32), CELT, Enhanced AC-3 (E-AC-3), EVS, FLAC, G.711, G.722, G.722.1, G.722.2 (AMR-WB), G.723.1, G.726, G.728, G.729, G.729.1, GSM-FR, HE-AAC, iLBC, iSAC, LA Lyra, Monkey's Audio, MP1, MP2 (MPEG-1, 2 Audio Layer II), MP3, Musepack, Nellymoser Asao, OptimFROG, Opus, Sac, Satin, SBC, SILK, Siren

7, Speex, SVOPC, True Audio (TTA), TwinVQ, USAC, Vorbis (Ogg), WavPack, and Windows Media Aud.

[0057] Video encoder 200 and video decoder 300 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 200 and video decoder 300 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder 200 and/or video decoder 300 may implement video encoder 200 and/or video decoder 300 in processing circuitry such as an integrated circuit and/or a microprocessor. Such a device may be a wireless communication device, such as a cellular telephone, or any other type of device described herein.

[0058] Video encoder 200 and video decoder 300 may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video coding extensions. Alternatively, video encoder 200 and video decoder 300 may operate according to other proprietary or industry standards, such as ITU-T H.266, also referred to as Versatile Video Coding (VVC). In other examples, video encoder 200 and video decoder 300 may operate according to a proprietary video codec/format, such as AOMedia Video 1 (AV1), extensions of AV1, and/or successor versions of AV1 (e.g., AV2). In other examples, video encoder 200 and video decoder 300 may operate according to other proprietary formats or industry standards. The techniques of this disclosure, however, are not limited to any particular coding standard or format. In general, video encoder 200 and video decoder 300 may be configured to perform the techniques of this disclosure in conjunction with any video coding techniques that use neural networks, such as NN-based video coding.

[0059] In general, video encoder 200 and video decoder 300 may perform block-based coding of pictures. The term “block” generally refers to a structure including data to be processed (e.g., encoded, decoded, or otherwise used in the encoding and/or decoding process). For example, a block may include a two-dimensional matrix of samples of

luminance and/or chrominance data. In general, video encoder 200 and video decoder 300 may code video data represented in a YUV (e.g., Y, Cb, Cr) format. That is, rather than coding red, green, and blue (RGB) data for samples of a picture, video encoder 200 and video decoder 300 may code luminance and chrominance components, where the chrominance components may include both red hue and blue hue chrominance components. In some examples, video encoder 200 converts received RGB formatted data to a YUV representation prior to encoding, and video decoder 300 converts the YUV representation to the RGB format. Alternatively, pre- and post-processing units (not shown) may perform these conversions.

[0060] This disclosure may generally refer to coding (e.g., encoding and decoding) of pictures to include the process of encoding or decoding data of the picture. Similarly, this disclosure may refer to coding of blocks of a picture to include the process of encoding or decoding data for the blocks, e.g., prediction and/or residual coding. An encoded video bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes) and partitioning of pictures into blocks. Thus, references to coding a picture or a block should generally be understood as coding values for syntax elements forming the picture or block.

[0061] HEVC defines various blocks, including coding units (CUs), prediction units (PUs), and transform units (TUs). According to HEVC, a video coder (such as video encoder 200) partitions a coding tree unit (CTU) into CUs according to a quadtree structure. That is, the video coder partitions CTUs and CUs into four equal, non-overlapping squares, and each node of the quadtree has either zero or four child nodes. Nodes without child nodes may be referred to as “leaf nodes,” and CUs of such leaf nodes may include one or more PUs and/or one or more TUs. The video coder may further partition PUs and TUs. For example, in HEVC, a residual quadtree (RQT) represents partitioning of TUs. In HEVC, PUs represent inter-prediction data, while TUs represent residual data. CUs that are intra-predicted include intra-prediction information, such as an intra-mode indication.

[0062] As another example, video encoder 200 and video decoder 300 may be configured to operate according to VVC. According to VVC, a video coder (such as video encoder 200) partitions a picture into a plurality of CTUs. Video encoder 200 may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure or Multi-Type Tree (MTT) structure. The QTBT structure removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT

structure includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to CUs.

[0063] In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and one or more types of triple tree (TT) (also called ternary tree (TT)) partitions. A triple or ternary tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple or ternary tree partition divides a block into three sub-blocks without dividing the original block through the center. The partitioning types in MTT (e.g., QT, BT, and TT), may be symmetrical or asymmetrical.

[0064] When operating according to the AV1 codec, video encoder 200 and video decoder 300 may be configured to code video data in blocks. In AV1, the largest coding block that can be processed is called a superblock. In AV1, a superblock can be either 128x128 luma samples or 64x64 luma samples. However, in successor video coding formats (e.g., AV2), a superblock may be defined by different (e.g., larger) luma sample sizes. In some examples, a superblock is the top level of a block quadtree. Video encoder 200 may further partition a superblock into smaller coding blocks. Video encoder 200 may partition a superblock and other coding blocks into smaller blocks using square or non-square partitioning. Non-square blocks may include $N/2 \times N$, $N \times N/2$, $N/4 \times N$, and $N \times N/4$ blocks. Video encoder 200 and video decoder 300 may perform separate prediction and transform processes on each of the coding blocks.

[0065] AV1 also defines a tile of video data. A tile is a rectangular array of superblocks that may be coded independently of other tiles. That is, video encoder 200 and video decoder 300 may encode and decode, respectively, coding blocks within a tile without using video data from other tiles. However, video encoder 200 and video decoder 300 may perform filtering across tile boundaries. Tiles may be uniform or non-uniform in size. Tile-based coding may enable parallel processing and/or multi-threading for encoder and decoder implementations.

[0066] In some examples, video encoder 200 and video decoder 300 may use a single QTBT or MTT structure to represent each of the luminance and chrominance components, while in other examples, video encoder 200 and video decoder 300 may use two or more QTBT or MTT structures, such as one QTBT/MTT structure for the luminance component and another QTBT/MTT structure for both chrominance components (or two QTBT/MTT structures for respective chrominance components).

[0067] Video encoder 200 and video decoder 300 may be configured to use quadtree partitioning, QTBT partitioning, MTT partitioning, superblock partitioning, or other partitioning structures.

[0068] In some examples, a CTU includes a coding tree block (CTB) of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate color planes and syntax structures used to code the samples. A CTB may be an $N \times N$ block of samples for some value of N such that the division of a component into CTBs is a partitioning. A component is an array or single sample from one of the three arrays (luma and two chroma) that compose a picture in 4:2:0, 4:2:2, or 4:4:4 color format or the array or a single sample of the array that compose a picture in monochrome format. In some examples, a coding block is an $M \times N$ block of samples for some values of M and N such that a division of a CTB into coding blocks is a partitioning.

[0069] The blocks (e.g., CTUs or CUs) may be grouped in various ways in a picture. As one example, a brick may refer to a rectangular region of CTU rows within a particular tile in a picture. A tile may be a rectangular region of CTUs within a particular tile column and a particular tile row in a picture. A tile column refers to a rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements (e.g., such as in a picture parameter set). A tile row refers to a rectangular region of CTUs having a height specified by syntax elements (e.g., such as in a picture parameter set) and a width equal to the width of the picture.

[0070] In some examples, a tile may be partitioned into multiple bricks, each of which may include one or more CTU rows within the tile. A tile that is not partitioned into multiple bricks may also be referred to as a brick. However, a brick that is a true subset of a tile may not be referred to as a tile. The bricks in a picture may also be arranged in a slice. A slice may be an integer number of bricks of a picture that may be exclusively contained in a single network abstraction layer (NAL) unit. In some examples, a slice includes either a number of complete tiles or only a consecutive sequence of complete bricks of one tile.

[0071] This disclosure may use “ $N \times N$ ” and “ N by N ” interchangeably to refer to the sample dimensions of a block (such as a CU or other video block) in terms of vertical and horizontal dimensions, e.g., 16×16 samples or 16 by 16 samples. In general, a 16×16 CU will have 16 samples in a vertical direction ($y = 16$) and 16 samples in a horizontal direction ($x = 16$). Likewise, an $N \times N$ CU generally has N samples in a vertical direction

and N samples in a horizontal direction, where N represents a nonnegative integer value. The samples in a CU may be arranged in rows and columns. Moreover, CUs need not necessarily have the same number of samples in the horizontal direction as in the vertical direction. For example, CUs may include NxM samples, where M is not necessarily equal to N.

[0072] Video encoder 200 encodes video data for CUs representing prediction and/or residual information, and other information. The prediction information indicates how the CU is to be predicted in order to form a prediction block for the CU. The residual information generally represents sample-by-sample differences between samples of the CU prior to encoding and the prediction block.

[0073] To predict a CU, video encoder 200 may generally form a prediction block for the CU through inter-prediction or intra-prediction. Inter-prediction generally refers to predicting the CU from data of a previously coded picture, whereas intra-prediction generally refers to predicting the CU from previously coded data of the same picture. To perform inter-prediction, video encoder 200 may generate the prediction block using one or more motion vectors. Video encoder 200 may generally perform a motion search to identify a reference block that closely matches the CU, e.g., in terms of differences between the CU and the reference block. Video encoder 200 may calculate a difference metric using a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations to determine whether a reference block closely matches the current CU. In some examples, video encoder 200 may predict the current CU using uni-directional prediction or bi-directional prediction.

[0074] Some examples of VVC also provide an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder 200 may determine two or more motion vectors that represent non-translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

[0075] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. Some examples of VVC provide sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder 200 selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the

left, or to the left of the current block in the same picture as the current block, assuming video encoder 200 codes CTUs and CUs in raster scan order (left to right, top to bottom).

[0076] Video encoder 200 encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder 200 may encode data representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder 200 may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

[0077] AV1 includes two general techniques for encoding and decoding a coding block of video data. The two general techniques are intra prediction (e.g., intra frame prediction or spatial prediction) and inter prediction (e.g., inter frame prediction or temporal prediction). In the context of AV1, when predicting blocks of a current frame of video data using an intra prediction mode, video encoder 200 and video decoder 300 do not use video data from other frames of video data. For most intra prediction modes, video encoder 200 encodes blocks of a current frame based on the difference between sample values in the current block and predicted values generated from reference samples in the same frame. Video encoder 200 determines predicted values generated from the reference samples based on the intra prediction mode.

[0078] Following prediction, such as intra-prediction or inter-prediction of a block, video encoder 200 may calculate residual data for the block. The residual data, such as a residual block, represents sample by sample differences between the block and a prediction block for the block, formed using the corresponding prediction mode. Video encoder 200 may apply one or more transforms to the residual block, to produce transformed data in a transform domain instead of the sample domain. For example, video encoder 200 may apply a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data. Additionally, video encoder 200 may apply a secondary transform following the first transform, such as a mode-dependent non-separable secondary transform (MDNSST), a signal dependent transform, a Karhunen-Loeve transform (KLT), or the like. Video encoder 200 produces transform coefficients following application of the one or more transforms.

[0079] As noted above, following any transforms to produce transform coefficients, video encoder 200 may perform quantization of the transform coefficients. Quantization generally refers to a process in which transform coefficients are quantized to possibly

reduce the amount of data used to represent the transform coefficients, providing further compression. By performing the quantization process, video encoder 200 may reduce the bit depth associated with some or all of the transform coefficients. For example, video encoder 200 may round an n -bit value down to an m -bit value during quantization, where n is greater than m . In some examples, to perform quantization, video encoder 200 may perform a bitwise right-shift of the value to be quantized.

[0080] Following quantization, video encoder 200 may scan the transform coefficients, producing a one-dimensional vector from the two-dimensional matrix including the quantized transform coefficients. The scan may be designed to place higher energy (and therefore lower frequency) transform coefficients at the front of the vector and to place lower energy (and therefore higher frequency) transform coefficients at the back of the vector. In some examples, video encoder 200 may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector, and then entropy encode the quantized transform coefficients of the vector. In other examples, video encoder 200 may perform an adaptive scan. After scanning the quantized transform coefficients to form the one-dimensional vector, video encoder 200 may entropy encode the one-dimensional vector, e.g., according to context-adaptive binary arithmetic coding (CABAC). Video encoder 200 may also entropy encode values for syntax elements describing metadata associated with the encoded video data for use by video decoder 300 in decoding the video data.

[0081] To perform CABAC, video encoder 200 may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether neighboring values of the symbol are zero-valued or not. The probability determination may be based on a context assigned to the symbol.

[0082] Video encoder 200 may further generate syntax data, such as block-based syntax data, picture-based syntax data, and sequence-based syntax data, to video decoder 300, e.g., in a picture header, a block header, a slice header, or other syntax data, such as a sequence parameter set (SPS), picture parameter set (PPS), or video parameter set (VPS). Video decoder 300 may likewise decode such syntax data to determine how to decode corresponding video data.

[0083] In this manner, video encoder 200 may generate a bitstream including encoded video data, e.g., syntax elements describing partitioning of a picture into blocks (e.g., CUs) and prediction and/or residual information for the blocks. Ultimately, video decoder 300 may receive the bitstream and decode the encoded video data.

[0084] In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information for partitioning of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0085] The residual information may be represented by, for example, quantized transform coefficients. Video decoder 300 may inverse quantize and inverse transform the quantized transform coefficients of a block to reproduce a residual block for the block. Video decoder 300 uses a signaled prediction mode (intra- or inter-prediction) and related prediction information (e.g., motion information for inter-prediction) to form a prediction block for the block. Video decoder 300 may then combine the prediction block and the residual block (on a sample-by-sample basis) to reproduce the original block. Video decoder 300 may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

[0086] This disclosure may generally refer to “signaling” certain information, such as syntax elements. The term “signaling” may generally refer to the communication of values for syntax elements and/or other data used to decode encoded video data. That is, video encoder 200 may signal values for syntax elements in the bitstream. In general, signaling refers to generating a value in the bitstream. As noted above, source device 102 may transport the bitstream to destination device 116 substantially in real time, or not in real time, such as might occur when storing syntax elements to storage device 112 for later retrieval by destination device 116.

[0087] In accordance with the techniques of this disclosure, a method of coding video data includes receiving a picture of video data, reconstructing the picture of video data, and performing a neural network (NN)-based filter process on the reconstructed picture of video data, wherein the NN-based filter process comprises performing a Canonical Polyadic (CP) decomposition with separable convolution.

[0088] Video coding standards include ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 or ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC), High Efficiency Video Coding (HEVC)

or ITU-T H.265, including its range extension, multiview extension (MV-HEVC) and scalable extension (SHVC). Versatile Video Coding (VVC) or ITU-T H.266 has recently been developed by the Joint Video Expert TEAM (JVET) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG).

[0089] The VVC specification has been finalized and referred as VVC FDIS hereinafter, is available from phenix.int-evry.fr/jvet/doc_end_user/documents/19_Teleconference/wg11/JVET-S2001-v17.

[0090] FIG. 2 is a conceptual diagram illustrating a hybrid video coding framework. Video coding standards since H.261 have been based on the so-called hybrid video coding principle, which is illustrated in FIG. 2. The term hybrid refers to the combination of two means to reduce redundancy in the video signal, i.e., prediction and transform coding with quantization of the prediction residual. Whereas prediction and transforms reduce redundancy in the video signal by decorrelation, quantization decreases the data of the transform coefficient representation by reducing their precision, ideally by removing only irrelevant details. This hybrid video coding design principle is also used in the two recent standards, ITU-T H.265/HEVC and ITU-T H.266/VVC.

[0091] As shown in FIG. 2, a modern hybrid video coder 130 generally performs block partitioning, motion-compensated or inter-picture prediction, intra-picture prediction, transformation, quantization, entropy coding, and post/in-loop filtering. In the example of FIG. 2, video coder 130 includes summation unit 134, transform unit 136, quantization unit 138, entropy coding unit 140, inverse quantization unit 142, inverse transform unit 144, summation unit 146, loop filter unit 148, decoded picture buffer (DPB) 150, intra prediction unit 152, inter-prediction unit 154, and motion estimation unit 156.

[0092] In general, video coder 130 may, when encoding video data, receive input video data 132. Block partitioning is used to divide a received picture (image) of the video data into smaller blocks for operation of the prediction and transform processes. Early video coding standards used a fixed block size, typically 16×16 samples. Recent standards, such as HEVC and VVC, employ tree-based partitioning structures to provide flexible partitioning.

[0093] Motion estimation unit 156 and inter-prediction unit 154 may predict input video data 132, e.g., from previously decoded data of DPB 150. Motion-compensated or inter-picture prediction takes advantage of the redundancy that exists between (hence “inter”) pictures of a video sequence. According to block-based motion compensation, which is used in the modern video codecs, the prediction is obtained from one or more previously

decoded pictures, i.e., the reference picture(s). The corresponding areas to generate the inter-prediction are indicated by motion information, including motion vectors and reference picture indices.

[0094] In a recent video codec, hierarchical prediction structures inside a group of pictures (GOP) is applied to improve coding efficiency. FIG. 3 illustrates an example hierarchical prediction structure 162 with a group of pictures (GOP) size equal to 16.

[0095] Referring again to FIG. 2, intra-picture prediction exploits the spatial redundancy that exists within a picture (hence "intra") by deriving the prediction for a block from already coded/decoded, spatially neighboring (reference) samples. The directional angular prediction, DC prediction and plane or planar prediction are used in the most recent video codec, including AVC, HEVC and VVC.

[0096] Hybrid video coding standards may apply a block transform to the prediction residual (regardless of whether it comes from inter- or intra-picture prediction). In early standards, including H.261/262/263, a discrete cosine transform (DCT) is employed. In HEVC and VVC, more transform kernel besides DCT are applied in order to account for different statistics in the specific video signal.

[0097] Quantization aims to reduce the precision of an input value or a set of input values in order to decrease the amount of data needed to represent the values. In hybrid video coding, the quantization is typically applied to individual transformed residual samples, e.g., to transform coefficients, resulting in integer coefficient levels. In recent video coding standards, the step size is derived from a so-called quantization parameter (QP) that controls the fidelity and bit rate. A larger step size lowers the bit rate but also deteriorates the quality, which e.g. results in video pictures exhibiting blocking artifacts and blurred details.

[0098] Summation unit 134 may calculate residual data as differences between input video data 132 and predicted data from intra prediction unit 152 or inter-prediction unit 154. Summation unit 134 provides residual blocks to transform unit 136, which applies one or more transforms to the residual block to generate transform blocks. Quantization unit 138 quantizes the transform blocks to form quantized transform coefficients. Entropy coding unit 140 entropy encodes the quantized transform coefficients, as well as other syntax elements, such as motion information or intra-prediction information, to generate output bitstream 158.

[0099] Meanwhile, inverse quantization unit 142 inverse quantizes the quantized transform coefficients, and inverse transform unit 144 inverse transforms the transform

coefficients, to reproduce residual blocks. Summation unit 146 combines the residual blocks with prediction blocks (on a sample-by-sample basis) to produce decoded blocks of video data. Loop filter unit 148 applies one or more filters (e.g., at least one of a neural network-based filter, a neural network-based loop filter, a neural network-based post loop filter, an adaptive in-loop filter, or a pre-defined adaptive in-loop filter) to the decoded block to produce filtered decoded blocks.

[0100] Entropy coding unit 140 may perform context-adaptive binary arithmetic coding (CABAC) on encoded video. CABAC is used in recent video codecs, e.g. AVC, HEVC and VVC, due to its high efficiency.

[0101] Loop filter unit 148 may perform post-loop or in-loop filtering. Post/In-Loop filtering is a filtering process (or combination of such processes) that is applied to the reconstructed picture to reduce the coding artifacts. The input of the filtering process is generally the reconstructed picture, which is the combination of the reconstructed residual signal (which includes quantization error) and the prediction. As shown in FIG. 2, the reconstructed pictures after in-loop filtering are stored in decoded picture buffer (DPB) 150 and are used as a reference for inter-picture prediction of subsequent pictures. The coding artifacts are mostly determined by the QP. Therefore, QP information is generally used in design of the filtering process. In HEVC, the in-loop filters include deblocking filtering and sample adaptive offset (SAO) filtering. In the VVC standard, an adaptive loop filter (ALF) was introduced as a third filter. The filtering process of ALF is as shown below:

$$R'(i, j) = R(i, j) + \left(\left(\sum_{k \neq 0} \sum_{l \neq 0} f(k, l) \times K(R(i + k, j + l) - R(i, j), c(k, l)) + 64 \right) \gg 7 \right), \quad (1)$$

where $R(i, j)$ is the samples before filtering process, $R'(i, j)$ is the sample value after filtering process. $f(k, l)$ denotes the filter coefficients, $K(x, y)$ is the clipping function and $c(k, l)$ denotes the clipping parameters. The variable k and l varies between $-\frac{L}{2}$ and $\frac{L}{2}$ where L denotes the filter length. The clipping function $K(x, y) = \min(y, \max(-y, x))$ which corresponds to the function $Clip3(-y, y, x)$. The clipping operation introduces non-linearity to make ALF more efficient by reducing the impact of neighbor sample values that are too different with the current sample value. In VVC, the filtering parameters can be signaled in the bit stream, and can be selected from pre-defined filter sets. The ALF filtering process can also be summarised with the following equation:

$$R'(i, j) = R(i, j) + ALF_residual_output(R), \quad (2)$$

[0102] FIG. 4 is a conceptual diagram illustrating a neural network-based filter 170 with four layers. Various studies have shown that embedding neural networks into, e.g., the hybrid video coding framework of FIG. 2, can improve compression efficiency. Neural networks have been used for intra prediction and inter prediction to improve the prediction efficiency. NN-based in-loop filtering is also a prominent research topic in recent years. In some examples, the filtering process is applied as a post-filter. In such examples, the filtering process is applied to the output picture, and the unfiltered picture may be used as reference picture.

[0103] NN-based filter 170 can be applied in addition to the existing filters, such as deblocking filters, sample adaptive offset (SAO), and/or adaptive loop filtering (ALF). NN-based filters can also be applied exclusively, where NN-based filters are designed to replace all of the existing filters. Additionally, or alternatively, NN-based filters, such as NN-based filter 170, may be designed to supplement, enhance, or replace any or all of the other filters.

[0104] The NN-based filtering process of FIG. 4 may take the reconstructed samples (e.g., luma and chroma samples which, in some examples, may be packed in a 3D volume with 6 planes) as inputs, and the intermediate outputs are residual samples, which are added back to the input to refine the input samples. The NN-based filter may use all color components (e.g., Y, U, and V, or Y, Cb, and Cr, e.g., luminance data 172A, blue-hue chrominance 172B, and red-hue chrominance 172C) as inputs 172 to exploit cross-component correlations. Different color components may share the same filter(s) (including network structure and model parameters) or each component may have its own specific filter(s).

[0105] The filtering process can also be generalized as follows:

$$R'(i, j) = R(i, j) + NN_filter_residual_output(R) \quad (3)$$

The model structure and model parameters of NN-based filter(s) can be pre-defined and be stored at video encoder 200 and video decoder 300. The filters can also be signalled in the bitstream.

[0106] In the example of FIG. 4, the NN-based filter may include a series of feature extraction layers, followed by an output convolution. In FIG. 4, the feature extraction layers may include a 3x3 convolution (conv) layer followed by a parametric rectified linear unit (PReLU) layer. The convolution layer applies a convolution operation to the input data, which involves a filter or kernel sliding over the input data (e.g., the reconstruction samples of input 172) and computing dot products at each position. The

convolution operation essentially captures local patterns within the input data. For example, in the context of image processing, these patterns could be edges, textures, or other visual features. The filter or kernel is a small matrix of weights that gets updated during the training process. By sliding this filter across the input data (or feature map from a previous layer) and computing the dot product at each position, the convolution layer creates a feature map that encodes spatial hierarchies and patterns detected in the input.

[0107] The output of a convolution layer is a set of feature maps, each corresponding to one filter, capturing different aspects of the input data. This layer helps the neural network to learn increasingly complex and abstract features as the data passes through deeper layers of the network. The first 3x3 in the nomenclature 3x3 conv 3x3x6x8 in FIG. 4 indicates that the convolution layer has a 3x3 filter size (e.g., a 3x3 matrix). 3x3x6x8 refers to both the input and output dimensions of the convolution layer, where 6 is the number of input channels, and 8 is the number of output channels.

[0108] The PReLU layer is an activation function used in neural networks, and was introduced as a variant of the ReLU (Rectified Linear Unit) activation function. As described above, the convolution layer outputs feature maps, each corresponding to one filter, representing detected features in the input. Following the convolution layer, the PReLU layer applies the PReLU activation function to each element of the feature maps produced by the convolution layer. For positive values, the PReLU layer acts like a standard ReLU, passing the value through. For negative values, instead of setting them to zero (e.g., as ReLU does), the PReLU layer allows a small, linear, negative output. This keeps the neurons active and maintains the gradient flow, which can be beneficial for learning in deep networks.

[0109] In summary, when a convolution layer is followed by a PReLU layer, the convolution layer first extracts features from the input data through a set of learned filters. The resulting feature maps are then passed through the PReLU activation function, which introduces non-linearity and helps to avoid the problem of dying neurons by allowing a small gradient when the inputs are negative. This combination is effective in learning complex patterns in the data while maintaining robust gradient flow, especially beneficial in deeper network architectures.

[0110] When NN-based filtering is applied in video coding, the whole video signal (pixel data) might be split into multiple processing units (e.g., 2D blocks), and each processing unit can be processed separately or be combined with other information

associated with this block of pixels. The possible choices of a processing unit include a frame, a slice/tile, a CTU, or any pre-defined or signaled shapes and sizes. Typically, NN-based filtering is performed on reconstructed blocks of video data. Here, reconstructed blocks and samples may refer to both decoded blocks produced by video decoder 300, as well blocks reconstructed in a reconstruction loop of video encoder 200.

[0111] To further improve the performance of NN-based filtering, different types of input data can be processed jointly to produce the filtered output. Input data may include, but is not limited to, reconstruction pixels/samples, prediction pixels/samples, pixels/samples after the loop filter(s), partitioning structure information, deblocking parameters (e.g., boundary strength (BS)), QP values, slice or picture types, or a filters applicability or coding modes map. Input data can be provided at different granularities. Luma reconstruction and prediction samples could be provided at the original resolution, whereas chroma samples could be provided at lower resolution, e.g. for 4:2:0 representation, or can be up-sampled to the Luma resolution to achieve per-pixel representation. Similarly, QP, BS, partitioning, or coding mode information can be provided at lower resolution, including cases with a single value per frame, slice or processing block (e.g., QP). In other examples, QP, BS, partitioning, or coding mode information can be expanded (e.g., replicated) to achieve per-pixel/sample representation.

[0112] An example of an architecture utilizing supplementary data is shown in FIG. 5. FIG. 5 is a block diagram illustrating an example CNN-based filter with padded input samples and supplementary data. NN-based filter 171 uses pixels/samples of the processing block combined with supplementary data as input 174. The input 174 may include 4 subblocks of interlaced luma samples ($Y \times 4$) 174A and associated blue hue chrominance (U) data 174B and red hue chrominance (V) data 174C. The supplementary data includes a quantization parameter (QP) step 176 and a boundary strength (BS) 178. The area of the input pixels/samples may be extended with 4 padded pixels/samples from each side. The resulting dimensions of the processing volume is $(4+64+4) \times (4+64+4) \times (4 Y+2UV+1QP + 3BS)$.

[0113] Relative to the NN-based filter in FIG. 4, NN-based filter 171 may include two or more hidden layers that utilize both 1×1 convolutions and a Leaky ReLU layer. A leaky ReLU layer. Similar to a PReLU layer, a Leaky ReLU layer allows a small, non-zero gradient to be output when the layer is not active. Instead of outputting zero for negative inputs, the Leaky ReLU multiplies these inputs by a small constant. This small

slope ensures that even neurons that would otherwise be inactive still contribute a small amount to the network's learning, reducing the likelihood of the dying ReLU problem.

[0114] To further improve the performance of NN-based filtering, multi-mode solutions can be designed. For example, for each processing unit, video encoder 200 may select among a set of modes based on rate-distortion optimization and the choice can be signaled in the bit-stream. The different modes may include different NN models, different values that are used as input information of the NN models, etc. In one example, video encoder 200 and video decoder 300 may use an NN-based filtering solution with multiple modes based on a single NN model by using different QP values as input of the NN model for different modes.

[0115] Examples of CNN ILF architectures that are being developed by JVET are now described. One such architecture is a multi-mode CNN ILF with an attention block. In JVET-Z0113, Y. Li, K. Zhang, L. Zhang, H. Wang, M. Coban, A. M. Kotra, M. Karczewicz, F. Galpin, K. Andersson, J. Ström, D. Liu, R. Sjöberg, EE1-1.7: Combined Test of EE1-1.6 and EE1-1.3, JVET-Z0113, April 2022 (hereinafter "JVET-Z0113"), an NN-based filtering solution with multiple modes is disclosed, as described above, with respect to the NN-based filtering with a multi-mode design.

[0116] In one example, an NN-based filtering solution with multiple modes may be used, as described above. The structure of the neural network is shown in FIG. 6. In the first part of FIG. 6, the different input data types are convolved with a number of kernels of size 3x3 (conv3x3) to produce feature maps, the convolved inputs undergo activation by parametric rectified linear units (PReLU), and then the results for each data type are concatenated (concat), fused, and subsampled once (transition) to create the output y. This output is then fed through attention residual (AttRes) blocks, each one with the structure shown in FIG. 6. The output from the last attention residual block z is fed into the last part of the network.

[0117] For example, the NN-based filter of FIG. 6 includes a first portion including input 3x3 convolutions 510A–510E and respective parametric rectified linear units (PReLU) 512A–512E for each of the inputs to generate feature maps (e.g., the feature extraction section of the NN-filter). Concatenation unit 514 concatenates the feature maps and provides them to fuse block 516 and transition block 522. The NN-based filter in FIG. 6 further includes a set 528 of attention residual (AttRes) blocks 530A–530N; and a last portion (e.g., the tail section) including 3x3 convolution 550, PReLU

552, 3x3 convolution 554, and pixel shuffle unit 556. The AttRes blocks may also be referred to as backbone blocks.

[0118] In the first portion (e.g., the feature extraction section), different inputs, including quantization parameter (QP) 500, partition information (part) 502, boundary strength (BS) 504, prediction samples (pred) 506, and reconstruction samples (rec) 508 are received. Respective 3x3 convolutions 510A–510E and PReLUs 512A–512E convolve and activate the respective inputs to produce feature maps. Concatenation unit 514 then concatenates the feature maps. Fuse block 516, including 1x1 convolution 518 and PReLU 520, fuses the concatenated feature maps. Transition block, including 3x3 convolution 524 and PReLU 526, subsamples the fused inputs to create output 188. Output 188 is then fed through set 528 of attention residual blocks 530A–530N, which may include a various number of attention residual blocks, e.g., 8. The attention block is explained further with respect to FIG. 7. Output 189 from the last of the set 528 of attention residual blocks 530 is fed to the last portion of the NN-based filter. In the last portion, which may be a tail block, 3x3 convolution 550, PReLU 552, 3x3 convolution 554, and pixel shuffle unit 556 processes output 189, and addition unit 558 combines this result with the original input reconstructions samples 508. This ultimately forms the filtered output for presentation and storage as reference for subsequent inter-prediction, e.g., in a decoded picture buffer (DPB). In some examples, the NN-based filter of FIG. 6 uses 96 feature maps.

[0119] FIG. 7 is a conceptual diagram illustrating an attention residual block of FIG. 6. That is, FIG. 7 depicts attention residual block 530, which may include components similar to those of attention residual blocks 530A–530N of FIG. 6. In this example, attention residual block 530 includes first 3x3 convolution 532, parametric rectified linear unit (PReLU) filter 534, second 3x3 convolution 536, an attention block 538, and addition unit 540. Addition unit 540 combines the output of attention block 538 and output 188, initially received by convolution 532, to generate output 189.

[0120] FIG. 8 is a conceptual diagram illustrating an example spatial attention layer of FIG. 7. As shown in FIG. 8, a spatial attention layer of attention residual block 530 includes 3x3 convolution 706, PReLU 708, 3x3 convolution 710, size expansion unit 712, 3x3 convolution 720, PReLU 722, and 3x3 convolution 724. 3x3 convolution 706 receives inputs 702, corresponding to quantization parameter (QP) 500, partition information (part) 502, boundary strength (BS) 504, prediction information (pred) 506, and reconstructed samples (rec) 508 of FIG. 6. 3x3 convolution 720 receives Z_K 704.

The outputs of size expansion unit 712 and 3x3 convolution 724 are combined, and then combined with R value 730 to generate S value 732. S value 732 is then combined with Z_K value 704 to generate output Z_{K+1} value 734.

[0121] Another example NN architecture is a reduced complexity multi-mode CNN ILF. In JVET-AC0155, S. Eadie, M. Coban, M. Karczewicz, EE1-1.9: Reduced complexity CNN-based in-loop filtering, JVET-AC0155, January 2023, an alternative design of the NN architecture was proposed. For example, a larger number of low complexity residual blocks in the backbone of the filter of the example of FIG. 6 may be used, along with a reduced number of channels (feature maps) and the removal of the attention modules. The proposed CNN-based filtering structure (e.g., for Luma filtering) is shown in FIG. 9.

[0122] FIG. 9 is a block diagram illustrating an example of a simplified CNN-based filter architecture with padded input samples and supplementary data. The NN-based filter of FIG. 9 includes 3x3 convolutions 810A–810E and PReLUs 812A–812E, which convolve corresponding inputs, i.e., QP 800, Part 802, BS 804, Pred 806, and Rec 808 to generate feature maps (e.g. the feature extraction section). Concatenation unit 814 concatenates the convolved inputs (e.g., the feature maps). Fuse block 816 then fuses the concatenated feature maps using 1x1 convolution 818 and PReLU 820. Transition block 822 then processes the fused data using 3x3 convolution 824 and PReLU 826.

[0123] In this example, the NN-based filter includes a set 828 of residual blocks 830A–830N (also called backbone blocks), each of which may be structured according to residual block structure 830 of FIG. 10, as discussed below. Residual blocks 830A–830N may replace AttRes blocks 530A–530N of FIG. 6. The example of FIG. 9 may be used for luminance (luma) filtering, although as discussed below, similar modifications may be made for chrominance (chroma) filtering.

[0124] The number of residual blocks and channels included in set 828 of FIG. 9 can be configured differently. That is, N may be set to a different value, and the number of channels in residual block structure 830 may be set to a number different than 160, to achieve different performance-complexity tradeoffs. Chroma filtering may be performed with these modifications for processing of chroma channels.

[0125] Set 828 of residual blocks 830A–830N has N instances of residual block structure 830. In one example, N may be equal to 32, such that there are 32 residual block structures. Residual blocks 830A–830N may use 64 feature maps, which is reduced relative to the 96 feature maps used in the example of FIG. 6.

[0126] In the last portion of FIG. 9, 3x3 convolution 850, PReLU 852, 3x3 convolution 854, and pixel shuffle unit 856 processes output of set 828, and addition unit 858 combines this result with the original input reconstructions samples (REC) 808. This ultimately forms the filtered output for presentation and storage as reference for subsequent inter-prediction, e.g., in a decoded picture buffer (DPB).

[0127] In one example, the number of residual blocks and feature maps (convolutions) used is $M=24$ and 64, respectively, in the example of FIG. 9. In residual blocks 830A–830N, the number of channels first goes up to 160 before the activation layer, and then goes down to 64 after the activation layer. The number of residual blocks and channels can be configured differently (M set to another value and the number of channels in the residual block can be set to a number different than 160) for different performance-complexity trade-offs. Chroma filtering may follow the concept of FIG. 6, with the above-described modifications to its backbone for processing of chroma channels.

[0128] In yet another NN architecture, the residual blocks can be replaced by filter blocks as shown in FIG. 11, discussed later in this disclosure. In this example, the bypass branch around convolution and activation layers in the residual block from the previous solution is removed. The number of channels and number of filter blocks may be configurable, for example, 64 channels, 24 filter blocks, with 160 channels before and after the activation, which results in the complexity of the network is 605.93kMAC and the number of parameters is 1.5M for the intra luma model.

[0129] Another example NN architecture is a multi-mode CNN ILF with separable convolution. Further complexity reduction of CCN ILF architecture may be achieved with utilization of the separable convolution in place of 2D convolutions (3x3). In JVET-AD0023, EE1 test 1.3.5, a low-rank convolution approximation decomposes a $3 \times 3 \times M \times N$ convolution into a pixel-wise convolution ($1 \times 1 \times M \times R$), two separable convolutions ($3 \times 1 \times R \times R$, $1 \times 3 \times R \times R$) and another pixel-wise convolution ($1 \times 1 \times R \times N$) were applied to a residual block of the architecture described in JVET-AC0155. In this example, R is the rank of the approximation, and can ablate the performance/complexity of the approximation.

[0130] FIG. 10 is a conceptual diagram illustrating an example residual block structure 830 of FIG. 9. In this example, residual block structure 830 includes first 1×1 convolution 832, which may increase a number of input channels to 160, before an activation layer (PReLU 834) processes the input channels. PReLU 834 may thereby reduce the number of channels to 64 through this processing. Second 1×1 convolution

836 then processes the reduced channels, followed by 3x3 convolution 838. Finally, combination unit 840 may combine the output of 3x3 convolution 838 with the original input received by residual block structure 830.

[0131] In yet another NN architecture, the residual blocks can be replaced by filter blocks (also called backbone blocks) as shown in FIG. 11. In this example, the bypass branch around all layers in each residual block is removed, as is shown FIG. 12. The number of channels and number of filter blocks is configurable, for example, 64 channels, 24 filter blocks, with 160 channels before and after the activation. In some examples, the complexity of such a network is 605.93kMAC and the number of parameters is 1.5M for an intra luma model.

[0132] FIG. 11 is a conceptual diagram illustrating another example filtering block structure that may be substituted for the set of attention residual blocks of FIG. 6 according to the techniques of this disclosure. The NN-based filter of FIG. 11 includes 3x3 convolutions 1010A–1010E and PReLUs 1012A–1012E, which convolve respective inputs, i.e., QP 1000, Part 1002, BS 1004, Pred 1006, and Rec 1008 to form feature maps (e.g., the feature extraction section). Concatenation unit 1014 concatenates the feature maps. Fuse block 1016 then fuses the concatenated inputs using 1x1 convolution 1018 and PReLU 1020. Transition block 1022 then processes the fused data using 3x3 convolution 1024 and PReLU 1026.

[0133] In this example, the NN-based filtering unit includes a set 1028 of N filter blocks 1030A–1030N (also called backbone blocks), each of which may have the structure of filter block 1030 of FIG. 12 as discussed below. Filter block structure 1030 may be substantially similar to residual block structure 830, except that combination unit 840 is omitted from filter block structure 1030, such that input is not combined with output. Instead, output of each residual block structure may be fed directly to the subsequent block.

[0134] In the last portion of FIG. 11, 3x3 convolution 1050, PReLU 1052, 3x3 convolution 1054, and pixel shuffle unit 1056 processes output of set 1028, and addition unit 1058 combines this result with the original input reconstructions samples (REC) 1008. This ultimately forms the filtered output for presentation and storage as reference for subsequent inter-prediction, e.g., in a decoded picture buffer (DPB).

[0135] In one example, the architecture of FIG. 11, is implemented with parameters K=64, M=160 and R=51, and total number of 24 residual blocks results in the complexity

of the network is 356.43kMAC and the number of parameters is 1.07M for the intra luma model.

[0136] FIG. 12 is a conceptual diagram illustrating an example filter block structure 1030 of FIG. 11. In this example, filter block structure 1030 includes first 1x1 convolution 1032, which may increase a number of input channels to 160, before an activation layer (PReLU 1034) processes the input channels. PReLU 1034 may thereby reduce the number of channels to 64 through this processing. Second 1x1 convolution 1036 then processes the reduced channels, followed by 3x3 convolution 1038. As discussed above, filter block structure 1030 does not include a combination unit, in contrast with the residual block structure 830 of FIG. 10.

[0137] Convolution with a 3x3 kernel is popular in NN-based filters. In the architectures described above, a 3X3xNxM convolution is utilized in multiple sections and blocks, with a 3x3 kernel sliding in the spatial (2D) domain. However, multi-dimensional convolution, such as a 2D kernel convolution, introduces significant complexity. In accordance with the techniques of this disclosure, video encoder 200 and video decoder 300 may be configured to utilize separable convolutions in place of, or instead of, a multi-dimensional convolution (e.g., a 3x3xNxM convolution). For example, two separable one-dimensional convolutions may be used in place of a 3x3 convolution in any section of an NN-based filter. The use of separable convolutions may reduce computation complexity and memory bandwidth requirements.

[0138] To avoid excessive computation and reduce parameter sets originating from multi-dimensional convolutions, such as 3x3 convolutions (or 2D convolution components of kernels of higher dimensionality) in the CNN-architectures described above or similar, this disclosure describes techniques where video encoder 200 and video decoder 300 are configured to utilize separable convolutions (e.g., 1D separable convolution) produced by low complexity approximation instead of multi-dimensional (e.g., 2D) convolutions sliding in the spatial direction. While the techniques of this disclosure are described with reference to 3x3 convolutions, the decomposition techniques of this disclosure may be used for any size of multi-dimensional convolutions (e.g., 4x4, 5x5 or larger). In general, a multi-dimensional convolution has a kernel size of $n1 \times n2$ in spatial dimension where $n1$ and $n2$ are positive integers. The values of $n1$ and $n2$ may be the same or different. The multi-dimensional convolution may further have a size of K in a depth dimension (e.g., $n1 \times n2 \times K$). In addition, with the

number of output channels M , the multi-dimensional convolution can be expressed as a 4-D tensor of $n_1 \times n_2 \times K \times M$.

[0139] In one example of the disclosure, a low-rank convolution approximation decomposes a $3 \times 3 \times M \times N$ convolution into a pixel-wise convolution ($1 \times 1 \times M \times R$), two separable convolutions ($3 \times 1 \times R \times R$, $1 \times 3 \times R \times R$), and another pixel-wise convolution ($1 \times 1 \times R \times N$). Here, R is the rank of the approximation, and can be used to adjust the performance/complexity of the approximation. The value of R may be an integer. In some examples, R can be derived as a function (ratio) of M or N , or $\max(M, N)$. In some examples, R can be set equal to $A * \max(M, N)$, with A being less than 1 (e.g., 0.2, 0.5, 0.8), A being higher than 1 (e.g., 1.0, 1.2), or other values.

[0140] In a general example, a multi-dimensional convolution may be approximated by a plurality of separable convolutions by performing a first convolution of size $n_1 \times 1$ and then performing a second convolution of size $1 \times n_2$ on the output of the first convolution.

[0141] FIG. 13 is a block diagram illustrating an example multiscale feature extraction backbone network with two-component convolution. The example of FIG. 13 may be similar to the example of FIG. 10 with the use of an approximation of a $3 \times 3 \times K \times K$ convolution with a $3 \times 1 \times K \times R$ convolution and a $1 \times 3 \times R \times K$ convolution.

[0142] In the example of FIG. 13, residual block 1420 includes a $1 \times 1 \times K \times M$ convolution 1402, followed by PReLU 1404. The output of PReLU 1404 is input to $1 \times 1 \times M \times K$ convolution 1406. A $3 \times 3 \times K \times K$ convolution 1408 of residual block 1420 is approximated by a $3 \times 1 \times K \times R$ convolution 1400 and then a $1 \times 3 \times R \times K$ convolution 1410. The output of $1 \times 3 \times R \times K$ convolution 1410 may be input to combination unit 1412 which may combine the output of $1 \times 3 \times R \times K$ convolution 1410 with an input to $1 \times 1 \times K \times M$ convolution 1402. R is the canonical rank of the decomposition. A lower rank implies a larger complexity reduction.

[0143] FIG. 14 is a block diagram illustrating an example unified filter with a joined model (joined luma and chroma). The multiscale feature extraction with a two-component convolution network illustrated in FIG. 14 is proposed in Y. Li, S. Eadie, D. Rusanovskyy, M. Karczewicz, EE1-Related: Combination test of EE1-1.3.5 and multiscale component of EE1-1.6, JVET-AD0211, April 2023 (JVET-AD0211). 3×3 convolutions may be decomposed into a $3 \times 1 \times C_1 \times R$ convolution and followed by a $1 \times 3 \times R \times C_2$ convolution, where C_1 and C_2 are the number of input and output channels, respectively, and R is the rank of the approximation (see V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, V. Lempitsky, Speeding up Convolution Neural Networks Using

Fine-tuned CP-Decomposition, ICLR 2015, Available online: <https://arxiv.org/pdf/1412.6553>). In some examples, the parameter R can be made proportional to $R=C1 \times C2 / (C1+C2)$ and controls the complexity of the approximation.

[0144] FIG. 14 shows a proposed architecture with 3x3 convolution blocks being replaced by separable convolutions of 3x1 and 1x3. In this example, residual block structure 1430 includes first 1x1 convolution 1432 before a first activation layer (PReLU 1434) and, in parallel with the first 1x1 convolution 1432 and PReLU 1434, a 3x3 convolution 1440 and a second activation layer (PReLU 1442). A second 1x1 convolution 1436 then processes the combined output of PReLU 1434 and PReLU 1442, followed by 3x3 convolution 1438. In the example of FIG. 14, however, 3x3 convolution 1440 may be approximated using a plurality of separable convolutions, shown as 3x1 convolution 1450 and 1x3 convolution 1452 in FIG. 14. Similarly, 3x3 convolution 1438 may be approximated using a plurality of separable convolutions, shown as 3x1 convolution 1460 and 1x3 convolution 1462 in FIG. 14.

[0145] As an example, the architecture illustrated in FIG. 14 may be implemented with parameters $R1=8$, $R2=44$, $M1=160$ and $M2=16$, and total number of 24 residual blocks, the complexity of the network will be 358.43kMAC and the number of parameters is 1.07M for the intra luma model.

[0146] FIG. 15 is a block diagram illustrating an example unified filter with a joined model (joined luma and chroma). The NN-based filter of FIG. 15 may include IPB 1500 (which provides information regarding whether a block is inter or intra predicted), QP_{SLICE} 1502 (which provides a quantization parameter for a slice), QP_{BASE} 1504 (which provides a quantization parameter for the sequence), boundary strength (BS) 1506, prediction samples (PRED) 1508, and reconstruction samples (REC) 1509. Video decoder 300 may apply a corresponding 3x3 convolution of 3x3 convolutions 1510A-1510F to each of the inputs. Video decoder 300 may apply a corresponding PReLU of PReLUs 1512A-1512F to the outputs of 3x3 convolutions 1510A-1510F. In some examples, the backbone blocks 1530A-1530N may be residual blocks.

[0147] Fuse block 1516 may fuse the feature maps using 1x1 convolution 1518 and PReLU 1520. Transition block 1522 then processes the fused data using 3x3 convolution 1524 and PReLU 1526.

[0148] Video decoder 300 may apply a set 1528 of backbone blocks. Set 1528 may include a plurality of filter blocks 1530A-1530N (also referred to herein as backbone blocks). In some examples, $N=24$, such that there are 24 filter blocks. Video decoder

300 may apply 3x3 convolution 1550 to the output of set 1528. Video decoder 300 may apply PReLU 1552 to the output of 3x3 convolution 1550. Video decoder 300 may apply 3x3 convolution 1554 to the output of PReLU 1552. Video decoder 300 may crop 1558 the output of 3x3 convolution 1554 to generate filtered reconstructed UV samples of the picture. Video decoder 300 may perform a pixel shuffle 1556 on the output of 3x3 convolution 1554 and crop 1560 the output of pixel shuffle 1556 to generate filtered reconstructed Y samples of the picture. For example, pixel shuffle 1556 may upsample its input such that the output of pixel shuffle 1556 has a size of $w*2, h*2, c/4$, where w is the width, h is the input height and c is the input number of channels of the input to pixel shuffle 1556. In some examples, the parameters of the example of FIG. 15 include one or more of: $d_1 = 192$, $d_2 = 32$, $d_3 = 16$, $d_4 = 16$, $d_5 = 16$, $C = 64$, and $d_6 = 48$.

[0149] FIG. 16 is a block diagram illustrating an example backbone block. Backbone block 1600 of FIG. 16 may be an example of any of filter blocks 1530A-1530N. Backbone block 1600 may include a multiscale branch 1620 where convolutions are performed in parallel. For example, a 1x1 convolution 1602 may be performed in parallel with a 3x1 convolution 1604 followed by a 1x3 convolution 1606. Multiscale branch 1620 may perform multiscale feature extraction.

[0150] Video decoder 300 may apply a PReLU 1608 to the output multiscale branch 1620. Video decoder 300 may apply a 1x1 convolution 1610 to the output of PReLU 1608. Video decoder 300 may apply a 1x3 convolution 1612 to the output of 1x1 convolution 1610. Video decoder 300 may apply a 3x1 convolution 1614 to the output of 1x3 convolution 1612. In some examples, the parameters of the example of FIG. 16 include $C = 64$, $C1 = 160$, $C21 = 32$, $C22 = 32$, and $C31 = 64$.

[0151] FIG. 17 is a block diagram illustrating example unified filter with separate luma and chroma models (luma). The NN-based filter of FIG. 17 may include IPB 1700, QP_{BASE} 1702, BS 1704, PRED 1706, and reconstructed luma samples ($REC_{EXT}Y$) 1708. Video decoder 300 may apply a corresponding 3x3 convolution of 3x3 convolutions 1710A-1710E to each of the inputs. Video decoder 300 may apply a corresponding PReLU of PReLUs 1712A-1712E to the outputs of 3x3 convolutions 1710A-1710E.

[0152] Fuse block 1716 may fuse the feature maps using 1x1 convolution 1718 and PReLU 1720. Transition block 1722 then processes the fused data using 3x3 convolution 1724 and PReLU 1726.

[0153] Video decoder 300 may apply a set 1728 of backbone blocks. Set 1728 may include a plurality of filter blocks 1730A-1730N, which may also be referred to as backbone blocks. In some examples, $N=20$, such that there are 20 filter blocks. Video decoder 300 may apply 3x3 convolution 1750 to the output of set 1728. Video decoder 300 may apply PReLU 1752 to the output of 3x3 convolution 1750. Video decoder 300 may apply 3x3 convolution 1754 to the output of PReLU 1752. Video decoder 300 may crop 1758 the output of 3x3 convolution 1754 to generate filtered reconstructed UV samples of the picture. Video decoder 300 may perform a pixel shuffle 1756 on the output of 3x3 convolution 1754 and crop 1760 the output of pixel shuffle 1756 to generate filtered reconstructed Y samples of the picture.

[0154] One or more of filter blocks 1730A-1730N of set 1728 may be examples of backbone block 1600 of FIG. 16. In some examples, the parameters of the example of FIG. 17 together with the example of FIG. 16 may include one or more of: $d_1 = 192$, $d_2 = 32$, $d_3 = 16$, $d_4 = 16$, $d_5 = 16$, $C = 64$, $C_1 = 160$, $C_{21} = 32$, $C_{22} = 32$, $C_{31} = 64$, $N = 20$, and $d_6 = 48$.

[0155] FIG. 18 is a block diagram illustrating an example unified filter with separate luma and chroma models (chroma). Inputs for the NN-based filter of FIG. 18 may include $REC_{EXT}Y$ 1800, which may be downsampled by a factor of 2, QP_{BASE} 1804, BS 1806, $PRED$ 1808, and reconstructed chroma samples ($REC_{EXT}UV$) 1809. Video decoder 300 may apply a corresponding 3x3 convolution of 3x3 convolutions 1810A-1810E to each of the inputs. Video decoder 300 may apply a corresponding PReLU of PReLUs 1812A-1812E to the outputs of 3x3 convolutions 1810A-1810E.

[0156] Fuse block 1816 may fuse the feature maps using 1x1 convolution 1818 and PReLU 1820. Transition block 1822 then processes the fused data using 3x3 convolution 1824 and PReLU 1826.

[0157] Video decoder 300 may apply a set 1828 of backbone blocks. Set 1828 may include a plurality of filter blocks 1830A-1830N, which may also be referred to as backbone blocks. In some examples, $N=16$, such that there are 24 filter blocks. Video decoder 300 may apply 3x3 convolution 1850 to the output of set 1828. Video decoder 300 may apply PReLU 1852 to the output of 3x3 convolution 1850. Video decoder 300 may apply 3x3 convolution 1854 to the output of PReLU 1852. Video decoder 300 may crop 1858 the output of 3x3 convolution 1854 to generate filtered reconstructed UV samples of the picture. Video decoder 300 may perform a pixel shuffle 1856 on the

output of 3x3 convolution 1854 and crop 1860 the output of pixel shuffle 1856 to generate filtered reconstructed Y samples of the picture.

[0158] One or more of filter blocks 1830A-1830N of set 1828 may use the backbone block architecture of FIG. 16. In some examples, the parameters of the example of FIG. 18 together with FIG. 16 may include one or more of: $d_1 = 192$, $d_2 = 32$, $d_3 = 16$, $d_4 = 16$, $d_5 = 16$, $C = 64$, $C_1 = 160$, $C_{21} = 32$, $C_{22} = 32$, $C_{31} = 64$, $N = 16$, and $d_6 = 48$.

[0159] A multiscale feature extraction backbone with two-component decomposition has been integrated into a unified model in EE. In addition, the specification from the EE includes two versions of the model, which are 1) a unified model for joined luma and chroma, as shown in FIG. 15, and 2) separate models for luma and chroma, respectively, as shown in FIGS. 17 and 18.

[0160] The Unified Filter (UF) Architecture adopted by JVET, (as shown in FIGS. 15-18), targets a relatively high-performance tier of video coding applications, operating at more than 400kMAC/pixel for coding of YCbCr 4:2:0 data. In order to run the in-loop filter with a video codec in state-of-the-art computationally and/or power constrained devices, the complexity of NN-based tools is constrained to be at levels lower than 20kMAC/pixel. Thus, the UF Architecture of FIGS. 15-18 cannot be implemented on such computationally and/or power constrained devices.

[0161] To achieve a required complexity level (e.g., <20kMAC/pixel) and preserve a relatively large part of coding performance afforded by the UF architecture, according to the techniques of this disclosure, a Canonical Polyadic (CP) decomposition with separable convolution is applied in the UF architecture to replace the 3x3 convolution in the backbone network and/or other block. The techniques of this disclosure also remove a multiscale feature extraction branch in a residue block, remove activation layers in a head block (e.g., input feature extraction), as well as employ other parameter adjustments.

[0162] FIG. 19 is a block diagram illustrating an example low complexity in-loop filter network architecture according to one or more aspects of this disclosure. For example, the architecture of FIG. 19 may be used to implement a low-complexity in-loop filter for joined YCbCr 4:2:0 processing. The example of FIG. 19 is also applicable for separate components Y,CbCr processing. The example of FIG. 19 may be a unified filter. A unified filter may include a filter having at least one quality input, such as a QP input, such that the filter may be used to handle different quality levels. As can be seen from FIG. 19, the example filter of FIG. 19 is a unified filter having both a QP_{SLICE} 1902 input

and QP_{BASE} 1904 input. A unified NN-based filter, such as that of FIG. 19, may be trained using various image and/or video data at varying levels of quality.

[0163] The architecture of FIG. 19 includes head block 1980, transition block 1922, set 1928 of one or more backbone blocks 1930, and tail block 1986. Inputs to head block 1980 include IPB 1900, QP_{SLICE} 1902, QP_{BASE} 1904, BS 1906, PRED 1908, and REC 1909.

[0164] Head block 1980 includes the blocks of the filter of FIG. 19 prior to the fuse block 1916. Head block 1980 includes 3x3 convolutions 1910A-1910F, which are applied to corresponding inputs. While shown as 3x3 convolutions, in some examples, any of, or any combination of 3x3 convolutions 1910A-1910F may be replaced by a CP decomposition, as is discussed further herein. Head block 1980 is shown with optional activation units 1912A-1912F. In some examples, no activation units are present in head block 1980. In some examples, one or more of activation units 1912A-1912F is a rectified linear unit (ReLU). In some examples, head block 1980 may include one or more multiscale branches (not shown for simplicity purposes), such as multiscale branch 1620 of FIG. 16.

[0165] Fuse block 1916 may fuse the feature maps using 1x1 convolution 1918 and PReLU 1920. Transition block 1922 may include blocks of filter 1980 occurring after fuse block 1916 and before any backbone or filter blocks (e.g., before filter blocks 1930A-1930N). Transition block 1922 processes the fused data using 3x3 convolution 1924 and PReLU 1926. In some examples, 3x3 convolution 1924 may be replaced by a CP decomposition, as discussed further herein. Set 1928 may include filter blocks 1930A-1930N, which may also be referred to as backbone blocks. In some examples, filter blocks 1930A-1930N may provide filtering functions of the filter of FIG. 19. In some examples, filter blocks 1930A-1930N may be located after transition block 1922 and prior to tail block 1986. In some examples, one or more of filter blocks 1930A-1930N may include a CP decomposition, as discussed further herein. In some examples, set 1928 (e.g., filter blocks 1930A-1930N) does not include a multiscale branch, such as multiscale branch 1620 of FIG. 16. Removal or omission of the multiscale branch may reduce complexity of the NN-based filter.

[0166] Tail block 1986 may include the blocks of the filter of FIG. 19 after all the backbone blocks or filter blocks (e.g., after filter blocks 1930A-1930N). Tail block 1986 may include a 3x3 convolution 1950, a PReLU 1952, a 3x3 convolution 1954, a crop unit 1958, a pixel shuffle unit 1956, and a crop unit 1960. In some examples, one or more of

3x3 convolution 1950 or 3x3 convolution 1954 may be replaced by a CP decomposition, as discussed further herein.

[0167] The parameters set forth herein for the example of FIG. 19 are only an example of the possible choices. The performance of such an architecture may vary with different configurations. In the example of FIG. 19, the following elements are described: activation in the head block; multiscale branch; and CP decomposition. It should be noted that all, any, or any combination of such elements may be utilized according to the techniques of this disclosure.

[0168] An example architecture and example of the parameters, such as number of inputs, features, channels and number of residue block is shown in FIG. 19. In some examples, the parameters for the example of FIG. 19 include one or more of: $d_1 = 12$, $d_2 = 8$, $d_3 = 4$, $d_4 = 2$, $d_5 = 2$, $C = 24$, $N = 11$, and $d_6 = 24$. The complexity is on the order of 19.55 kMAC for the illustrated configuration.

[0169] For example, with respect to activation in head block 1980, in some examples, activation units 1912A-1912F, which may be PReLU in the current UF architecture, are removed for a group of inputs. Removal of PReLU may reduce complexity of the NN-based filter. For example, head block 1980 may not include a corresponding PReLU for at least one input to head block 1980. In some examples, any of, or any combination of, activation units 1912A-1912E may be removed from head block 1980. For example, head block 1980 may not include a corresponding activation layer for at least one input to head block 1980. In some examples, one or more of activation units 1912A-1912F may be a rectified linear unit ReLU. Replacement of a PReLU with a ReLU may reduce complexity of the NN-based filter. For example, one or more of the PReLU in the current UF architecture may be replaced by a ReLU. In some examples, activation units 1912A-1912F for head block 1980 are removed to allow only linear operation(s) to be performed before the fusing operation. For example, the outputs of one or more of the 3x3 convolutions 1910A-1910F are input to 1x1 convolution 1918 without passing through any activation layers. Removal of activation layers may reduce complexity of the NN-based filter.

[0170] For example, with respect to multiscale branches, in some examples, the multiscale branch of the current UF architecture (e.g., as shown in FIG. 16) is removed from the backbone residue blocks (e.g., filter blocks 1930A-1930N) such that the backbone blocks do not perform parallel convolutions. Removal or omission of the multiscale branch may reduce complexity of the NN-based filter. In some examples, the

multiscale branch is moved from the backbone blocks (e.g., a multiscale branch is shown in FIG. 16 in a backbone block) to head block 1980 to facilitate the multiscale feature extraction at a relatively low computational cost (e.g., lower than if the multiscale branch was implemented in the backbone blocks). For example, head block 1980 may include at least one multiscale branch similar to multiscale branch 1620. In some examples, the multiscale branch may be applied to REC 1909, e.g., prior to 3x3 convolution 1910F.

[0171] For example, with respect to CP decomposition, a CP decomposition may be used to decompose multi-dimensional convolutions, for example, by using 1) a 1x1 convolution, and 2) plurality of separate convolutions in a horizontal dimension and in a vertical dimension (e.g., a 1x3 convolution and a 3x1 convolution), followed by 3) a 1x1 convolution to combine the extracted features from the previous layers. In some examples, the first 1x1 convolution of the CP decomposition may be combined with a previous layer, therefore, the first 1x1 convolution may be skipped. For example, the CP decomposition may include at least one 1x1 convolution and separable 3x1 convolutions in a horizontal and a vertical dimension. For example, the separate convolutions in the horizontal dimension and the vertical dimension may include a convolution in the horizontal dimension and a convolution in the vertical dimension. The convolution in the horizontal dimension may include one of a 1x3 convolution or a 3x1 convolution. The convolution in the vertical dimension may include the other of the 1x3 convolution or the 3x1 convolution. In some examples, the at least one 1x1 convolution includes a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension, and a second 1x1 convolution occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension. In some examples, the first 1x1 convolutions may be omitted if there is no expansion or contraction of channels. In some examples, if the convolution kernel size is NxM, the spatial decomposition of the kernel may be approximated by a Nx1 convolution followed by 1xM convolution.

[0172] In some examples, video encoder 200 or video decoder 300 may employ separable decomposition (e.g., CP decomposition) in set 1928 and/or tail block 1986. In some examples, video encoder 200 or video decoder 300 may apply the separable decomposition (e.g., CP decomposition) to all the blocks (e.g., head block 1980, transition block 1922, set 1928, and tail block 1986). For example, a CP decomposition may be specific type of separable decomposition and may use a plurality of convolutions to

approximate a higher order multidimensional convolution. For example, a CP decomposition may approximate a 3x3 convolution by using a 1x1 convolution, a 1x3 convolution, a 3x1 convolution, and another 1x1 convolution.

[0173] FIG. 20 is a block diagram illustrating an example backbone block according to one or more aspects of this disclosure. Backbone block 2000 may be an example of any of filter blocks 1930A-1930N of set 1928 of FIG. 19. Backbone block 2000 differs from backbone block 1600 of the UF in that backbone block 2000 does not include a multiscale branch like backbone block 1600 and that the multidimensional convolution replaced by the convolutions 1610, 1612, and 1614, are in backbone block 2000 replaced by a CP decomposition. The CP decomposition in backbone block 2000 includes a 1x1 convolution 2006, a 1x3 convolution 2008, a 3x1 convolution 2010, and a 1x1 convolution 2012. In some examples, the parameters of FIG. 20 include $C = 24$, $C1 = 72$, and $C21 = 24$.

[0174] FIG. 21 is a block diagram illustrating an example CP decomposition according to one or more aspects of this disclosure. CP decomposition 2100 may include a 1x1 convolution 2106, a 1x3 convolution 2108, a 3x1 convolution 2110, and a 1x1 convolution 2112. It should be understood that CP decomposition 2100 may replace any of, or any combination of, 3x3 convolutions 1910A-1910F of head block 1980, 3x3 convolution 1924 of transition block 1922, 3x3 convolution 1950 and/or 3x3 convolution 1954 of tail block 1986. In some examples, the parameters of FIG. 21 include $C = 24$, $C1 = 72$, and $C21 = 24$.

[0175] The techniques described herein may be applicable to in-loop filters with a wide range of complexity by choosing different parameter configurations. The techniques of this disclosure may reduce computation complexity and memory bandwidth requirements and provide competitive performance, such as for devices with complexity and/or memory bandwidth constraints. Examples described in this document are related to NN-based in-loop filtering, however, such examples may be applicable to any NN-based video coding tool that consumes input data with certain statistical properties, such as static content or sparse representation.

[0176] FIG. 22 is a flow diagram illustrating example NN-based in-loop filtering techniques according to one or more aspects of this disclosure. Video decoder 300 may receive data for reconstructing a picture of video data (2200). For example, video encoder 200 may transmit encoded video data of the picture in a bitstream to video decoder 300 and video decoder 300 may receive the encoded video data of the picture in the bitstream.

[0177] Video decoder 300 may reconstruct the picture of video data (2202). For example, video decoder 300 may decode the encoded video data to reconstruct the picture of video data.

[0178] Video decoder 300 may apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution (2204). For example, video decoder 300 may filter the reconstructed picture using an NN-based filter, such as that of FIG. 19. The NN-based filter may include head block 1980, transition block 1922, one or more backbone blocks (e.g., set 1928), and tail block 1986. At least one of head block 1980, transition block 1922, one or more backbone blocks, or tail block 1986 includes a CP decomposition with separable convolution. The one or more backbone blocks may be residual blocks.

[0179] For example, head block 1980 may include the blocks of the NN-based filter 19 prior to a fuse block where feature maps are fused. Transition block 1922 may include blocks of the NN-based filter occurring after the feature maps are fused (e.g., after the fuse block and therefore after the head block) and before any backbone or filter blocks. Backbone blocks (e.g., filter blocks 1930A-1930N) may provide filter functions and include blocks of the NN-based filter between transition block 1922 and tail block 1986. Tail block 1986 may include blocks of the NN-based filter occurring after all of the backbone blocks.

[0180] In some examples, the CP decomposition includes at least one 1x1 convolution, a convolution in a horizontal dimension and a convolution in a vertical dimension. In some examples, the convolution in the horizontal dimension comprises one of a 1x3 convolution or a 3x1 convolution, and wherein the convolution in the vertical dimension includes another of the 1x3 convolution or the 3x1 convolution. For example, the CP decomposition may include at least one 1x1 convolution, a 1x3 convolution, and a 3x1 convolution. In some examples, the at least one 1x1 convolution includes a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension and a second 1x1 convolution occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension. It should be noted that the order of the convolutions may be changed and the two linear

layers may be combined. In some examples, the second 1x1 convolution combines extracted features from previous layers.

[0181] In some examples, one or more backbone blocks and tail block 1986 each include a corresponding CP decomposition with separable convolution. In other words, the one or more backbone blocks includes a CP decomposition with separable convolution and the tail block includes a CP decomposition with separable convolution. In some examples, head block 1980, transition block 1922, one or more backbone blocks 1930, and tail block 1986 each comprise a corresponding CP decomposition with separable convolution. In some examples, one or more backbone blocks 1930 do not include a multiscale branch. For example, a multiscale branch may perform multiscale feature extraction through the use of parallel convolutions such as is shown in FIG. 16.

[0182] In some examples, head block 1980 does not include a corresponding parametric rectified linear unit activation (PReLU) for at least one input to head block 1980. In some examples, head block 1980 does not include a corresponding activation layer for at least one input to the head block. In some examples, head block 1980 does not include any activation layers. In some examples, head block 1980 includes at least one corresponding rectified linear unit activation (ReLU) for at least one input to head block 1980. In some examples, head block 1980 includes at least one multiscale branch. In some examples, the one or more backbone blocks of the NN-based filter each include 24 channels. For example, $C=24$.

[0183] FIG. 23 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 23 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 200 according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video encoding devices that are configured to other video coding standards and video coding formats, such as AV1 and successors to the AV1 video coding format.

[0184] In the example of FIG. 23, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and entropy encoding unit 220. Any or all of video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208,

inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, DPB 218, and entropy encoding unit 220 may be implemented in one or more processors or in processing circuitry. For instance, the units of video encoder 200 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video encoder 200 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0185] Video data memory 230 may store video data to be encoded by the components of video encoder 200. Video encoder 200 may receive the video data stored in video data memory 230 from, for example, video source 104 (FIG. 1). DPB 218 may act as a reference picture memory that stores reference video data for use in prediction of subsequent video data by video encoder 200. Video data memory 230 and DPB 218 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory 230 and DPB 218 may be provided by the same memory device or separate memory devices. In various examples, video data memory 230 may be on-chip with other components of video encoder 200, as illustrated, or off-chip relative to those components.

[0186] In this disclosure, reference to video data memory 230 should not be interpreted as being limited to memory internal to video encoder 200, unless specifically described as such, or memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as reference memory that stores video data that video encoder 200 receives for encoding (e.g., video data for a current block that is to be encoded). Memory 106 of FIG. 1 may also provide temporary storage of outputs from the various units of video encoder 200.

[0187] The various units of FIG. 23 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software

instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0188] Video encoder 200 may include arithmetic logic units (ALUs), elementary function units (EFUs), digital circuits, analog circuits, and/or programmable cores, formed from programmable circuits. In examples where the operations of video encoder 200 are performed using software executed by the programmable circuits, memory 106 (FIG. 1) may store the instructions (e.g., object code) of the software that video encoder 200 receives and executes, or another memory within video encoder 200 (not shown) may store such instructions.

[0189] Video data memory 230 is configured to store received video data. Video encoder 200 may retrieve a picture of the video data from video data memory 230 and provide the video data to residual generation unit 204 and mode selection unit 202. Video data in video data memory 230 may be raw video data that is to be encoded.

[0190] Mode selection unit 202 includes a motion estimation unit 222, a motion compensation unit 224, and an intra-prediction unit 226. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes. As examples, mode selection unit 202 may include a palette unit, an intra-block copy unit (which may be part of motion estimation unit 222 and/or motion compensation unit 224), an affine unit, a linear model (LM) unit, or the like.

[0191] Mode selection unit 202 generally coordinates multiple encoding passes to test combinations of encoding parameters and resulting rate-distortion values for such combinations. The encoding parameters may include partitioning of CTUs into CUs, prediction modes for the CUs, transform types for residual data of the CUs, quantization parameters for residual data of the CUs, and so on. Mode selection unit 202 may ultimately select the combination of encoding parameters having rate-distortion values that are better than the other tested combinations.

[0192] Video encoder 200 may partition a picture retrieved from video data memory 230 into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit 202 may partition a CTU of the picture in accordance with a tree structure, such as the MTT structure, QTBT structure, superblock structure, or the quad-tree structure described above. As described above, video encoder 200 may form one or more CUs

from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a “video block” or “block.”

[0193] In general, mode selection unit 202 also controls the components thereof (e.g., motion estimation unit 222, motion compensation unit 224, and intra-prediction unit 226) to generate a prediction block for a current block (e.g., a current CU, or in HEVC, the overlapping portion of a PU and a TU). For inter-prediction of a current block, motion estimation unit 222 may perform a motion search to identify one or more closely matching reference blocks in one or more reference pictures (e.g., one or more previously coded pictures stored in DPB 218). In particular, motion estimation unit 222 may calculate a value representative of how similar a potential reference block is to the current block, e.g., according to sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or the like. Motion estimation unit 222 may generally perform these calculations using sample-by-sample differences between the current block and the reference block being considered. Motion estimation unit 222 may identify a reference block having a lowest value resulting from these calculations, indicating a reference block that most closely matches the current block.

[0194] Motion estimation unit 222 may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit 222 may then provide the motion vectors to motion compensation unit 224. For example, for uni-directional inter-prediction, motion estimation unit 222 may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit 222 may provide two motion vectors. Motion compensation unit 224 may then generate a prediction block using the motion vectors. For example, motion compensation unit 224 may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit 224 may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit 224 may retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

[0195] When operating according to the AV1 video coding format, motion estimation unit 222 and motion compensation unit 224 may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion

compensation, affine motion compensation, overlapped block motion compensation (OBMC), and/or compound inter-intra prediction.

[0196] As another example, for intra-prediction, or intra-prediction coding, intra-prediction unit 226 may generate the prediction block from samples neighboring the current block. For example, for directional modes, intra-prediction unit 226 may generally mathematically combine values of neighboring samples and populate these calculated values in the defined direction across the current block to produce the prediction block. As another example, for DC mode, intra-prediction unit 226 may calculate an average of the neighboring samples to the current block and generate the prediction block to include this resulting average for each sample of the prediction block.

[0197] When operating according to the AV1 video coding format, intra-prediction unit 226 may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, chroma-from-luma (CFL) prediction, intra block copy (IBC), and/or color palette mode. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes.

[0198] Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, unencoded version of the current block from video data memory 230 and the prediction block from mode selection unit 202. Residual generation unit 204 calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit 204 may also determine differences between sample values in the residual block to generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit 204 may be formed using one or more subtractor circuits that perform binary subtraction.

[0199] In examples where mode selection unit 202 partitions CUs into PUs, each PU may be associated with a luma prediction unit and corresponding chroma prediction units. Video encoder 200 and video decoder 300 may support PUs having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction unit of the PU. Assuming that the size of a particular CU is $2N \times 2N$, video encoder 200 may support PU sizes of $2N \times 2N$ or $N \times N$ for intra prediction, and symmetric PU sizes of $2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$, or similar for inter prediction. Video encoder 200 and video decoder 300

may also support asymmetric partitioning for PU sizes of $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$ for inter prediction.

[0200] In examples where mode selection unit 202 does not further partition a CU into PUs, each CU may be associated with a luma coding block and corresponding chroma coding blocks. As above, the size of a CU may refer to the size of the luma coding block of the CU. The video encoder 200 and video decoder 300 may support CU sizes of $2N \times 2N$, $2N \times N$, or $N \times 2N$.

[0201] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as some examples, mode selection unit 202, via respective units associated with the coding techniques, generates a prediction block for the current block being encoded. In some examples, such as palette mode coding, mode selection unit 202 may not generate a prediction block, and instead generate syntax elements that indicate the manner in which to reconstruct the block based on a selected palette. In such modes, mode selection unit 202 may provide these syntax elements to entropy encoding unit 220 to be encoded.

[0202] As described above, residual generation unit 204 receives the video data for the current block and the corresponding prediction block. Residual generation unit 204 then generates a residual block for the current block. To generate the residual block, residual generation unit 204 calculates sample-by-sample differences between the prediction block and the current block.

[0203] Transform processing unit 206 applies one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a “transform coefficient block”). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit 206 may apply a discrete cosine transform (DCT), a directional transform, a Karhunen-Loeve transform (KLT), or a conceptually similar transform to a residual block. In some examples, transform processing unit 206 may perform multiple transforms to a residual block, e.g., a primary transform and a secondary transform, such as a rotational transform. In some examples, transform processing unit 206 does not apply transforms to a residual block.

[0204] When operating according to AV1, transform processing unit 206 may apply one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a “transform coefficient block”). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block.

For example, transform processing unit 206 may apply a horizontal/vertical transform combination that may include a discrete cosine transform (DCT), an asymmetric discrete sine transform (ADST), a flipped ADST (e.g., an ADST in reverse order), and an identity transform (IDTX). When using an identity transform, the transform is skipped in one of the vertical or horizontal directions. In some examples, transform processing may be skipped.

[0205] Quantization unit 208 may quantize the transform coefficients in a transform coefficient block, to produce a quantized transform coefficient block. Quantization unit 208 may quantize transform coefficients of a transform coefficient block according to a quantization parameter (QP) value associated with the current block. Video encoder 200 (e.g., via mode selection unit 202) may adjust the degree of quantization applied to the transform coefficient blocks associated with the current block by adjusting the QP value associated with the CU. Quantization may introduce loss of information, and thus, quantized transform coefficients may have lower precision than the original transform coefficients produced by transform processing unit 206.

[0206] Inverse quantization unit 210 and inverse transform processing unit 212 may apply inverse quantization and inverse transforms to a quantized transform coefficient block, respectively, to reconstruct a residual block from the transform coefficient block. Reconstruction unit 214 may produce a reconstructed block corresponding to the current block (albeit potentially with some degree of distortion) based on the reconstructed residual block and a prediction block generated by mode selection unit 202. For example, reconstruction unit 214 may add samples of the reconstructed residual block to corresponding samples from the prediction block generated by mode selection unit 202 to produce the reconstructed block.

[0207] Filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. Operations of filter unit 216 may be skipped, in some examples.

[0208] When operating according to AV1, filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. In other examples, filter unit 216 may apply a constrained directional enhancement filter (CDEF), which may be applied after deblocking, and may include the application of non-separable, non-linear, low-pass directional filters based on estimated edge directions. Filter unit 216 may also

include a loop restoration filter, which is applied after CDEF, and may include a separable symmetric normalized Wiener filter or a dual self-guided filter.

[0209] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not performed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are performed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

[0210] In general, entropy encoding unit 220 may entropy encode syntax elements received from other functional components of video encoder 200. For example, entropy encoding unit 220 may entropy encode quantized transform coefficient blocks from quantization unit 208. As another example, entropy encoding unit 220 may entropy encode prediction syntax elements (e.g., motion information for inter-prediction or intra-mode information for intra-prediction) from mode selection unit 202. Entropy encoding unit 220 may perform one or more entropy encoding operations on the syntax elements, which are another example of video data, to generate entropy-encoded data. For example, entropy encoding unit 220 may perform a context-adaptive variable length coding (CAVLC) operation, a CABAC operation, a variable-to-variable (V2V) length coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation on the data. In some examples, entropy encoding unit 220 may operate in bypass mode where syntax elements are not entropy encoded.

[0211] Video encoder 200 may output a bitstream that includes the entropy encoded syntax elements needed to reconstruct blocks of a slice or picture. In particular, entropy encoding unit 220 may output the bitstream.

[0212] In accordance with AV1, entropy encoding unit 220 may be configured as a symbol-to-symbol adaptive multi-symbol arithmetic coder. A syntax element in AV1 includes an alphabet of N elements, and a context (e.g., probability model) includes a set of N probabilities. Entropy encoding unit 220 may store the probabilities as n-bit (e.g., 15-bit) cumulative distribution functions (CDFs). Entropy encoding unit 220 may

perform recursive scaling, with an update factor based on the alphabet size, to update the contexts.

[0213] The operations described above are described with respect to a block. Such description should be understood as being operations for a luma coding block and/or chroma coding blocks. As described above, in some examples, the luma coding block and chroma coding blocks are luma and chroma components of a CU. In some examples, the luma coding block and the chroma coding blocks are luma and chroma components of a PU.

[0214] In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-prediction process may be the same for the luma coding block and the chroma coding blocks.

[0215] Video encoder 200 represents an example of a device configured to encode video data including one or more memories configured to store video data, and one or more processors configured to receive a picture of video data, reconstruct the picture of video data, and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0216] FIG. 24 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 24 is provided for purposes of explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder 300 according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0217] In the example of FIG. 24, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310,

filter unit 312, and DPB 314. Any or all of CPB memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314 may be implemented in one or more processors or in processing circuitry. For instance, the units of video decoder 300 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video decoder 300 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0218] Prediction processing unit 304 includes motion compensation unit 316 and intra-prediction unit 318. Prediction processing unit 304 may include additional units to perform prediction in accordance with other prediction modes. As examples, prediction processing unit 304 may include a palette unit, an intra-block copy unit (which may form part of motion compensation unit 316), an affine unit, a linear model (LM) unit, or the like. In other examples, video decoder 300 may include more, fewer, or different functional components.

[0219] When operating according to AV1, motion compensation unit 316 may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion compensation, affine motion compensation, OBMC, and/or compound inter-intra prediction, as described above. Intra-prediction unit 318 may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, CFL, IBC, and/or color palette mode, as described above.

[0220] CPB memory 320 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 300. The video data stored in CPB memory 320 may be obtained, for example, from computer-readable medium 110 (FIG. 1). CPB memory 320 may include a CPB that stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory 320 may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder 300. DPB 314 generally stores decoded pictures, which video decoder 300 may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory 320 and DPB 314 may be formed by any of a variety of memory devices, such as DRAM, including SDRAM, MRAM, RRAM, or other types of memory devices. CPB memory 320 and DPB 314 may be provided by the same memory device or separate memory

devices. In various examples, CPB memory 320 may be on-chip with other components of video decoder 300, or off-chip relative to those components.

[0221] Additionally or alternatively, in some examples, video decoder 300 may retrieve coded video data from memory 120 (FIG. 1). That is, memory 120 may store data as discussed above with CPB memory 320. Likewise, memory 120 may store instructions to be executed by video decoder 300, when some or all of the functionality of video decoder 300 is implemented in software to be executed by processing circuitry of video decoder 300.

[0222] The various units shown in FIG. 24 are illustrated to assist with understanding the operations performed by video decoder 300. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. 23, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

[0223] Video decoder 300 may include ALUs, EFUs, digital circuits, analog circuits, and/or programmable cores formed from programmable circuits. In examples where the operations of video decoder 300 are performed by software executing on the programmable circuits, on-chip or off-chip memory may store instructions (e.g., object code) of the software that video decoder 300 receives and executes.

[0224] Entropy decoding unit 302 may receive encoded video data from the CPB and entropy decode the video data to reproduce syntax elements. Prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, and filter unit 312 may generate decoded video data based on the syntax elements extracted from the bitstream.

[0225] In general, video decoder 300 reconstructs a picture on a block-by-block basis. Video decoder 300 may perform a reconstruction operation on each block individually

(where the block currently being reconstructed, i.e., decoded, may be referred to as a “current block”).

[0226] Entropy decoding unit 302 may entropy decode syntax elements defining quantized transform coefficients of a quantized transform coefficient block, as well as transform information, such as a quantization parameter (QP) and/or transform mode indication(s). Inverse quantization unit 306 may use the QP associated with the quantized transform coefficient block to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit 306 to apply. Inverse quantization unit 306 may, for example, perform a bitwise left-shift operation to inverse quantize the quantized transform coefficients. Inverse quantization unit 306 may thereby form a transform coefficient block including transform coefficients.

[0227] After inverse quantization unit 306 forms the transform coefficient block, inverse transform processing unit 308 may apply one or more inverse transforms to the transform coefficient block to generate a residual block associated with the current block. For example, inverse transform processing unit 308 may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the transform coefficient block.

[0228] Furthermore, prediction processing unit 304 generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit 302. For example, if the prediction information syntax elements indicate that the current block is inter-predicted, motion compensation unit 316 may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB 314 from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference picture relative to the location of the current block in the current picture. Motion compensation unit 316 may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit 224 (FIG. 23).

[0229] As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit 318 may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit 318 may generally perform the intra-prediction process in a manner that is substantially similar to that described with respect

to intra-prediction unit 226 (FIG. 23). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0230] Reconstruction unit 310 may reconstruct the current block using the prediction block and the residual block. For example, reconstruction unit 310 may add samples of the residual block to corresponding samples of the prediction block to reconstruct the current block.

[0231] Filter unit 312 may perform one or more filter operations on reconstructed blocks. For example, filter unit 312 may perform deblocking operations to reduce blockiness artifacts along edges of the reconstructed blocks. Operations of filter unit 312 are not necessarily performed in all examples.

[0232] Video decoder 300 may store the reconstructed blocks in DPB 314. For instance, in examples where operations of filter unit 312 are not performed, reconstruction unit 310 may store reconstructed blocks to DPB 314. In examples where operations of filter unit 312 are performed, filter unit 312 may store the filtered reconstructed blocks to DPB 314. As discussed above, DPB 314 may provide reference information, such as samples of a current picture for intra-prediction and previously decoded pictures for subsequent motion compensation, to prediction processing unit 304. Moreover, video decoder 300 may output decoded pictures (e.g., decoded video) from DPB 314 for subsequent presentation on a display device, such as display device 118 of FIG. 1.

[0233] In this manner, video decoder 300 represents an example of a video decoding device including one or more memories configured to store video data, and one or more processors configured to receive a picture of video data, reconstruct the picture of video data, and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0234] FIG. 25 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 23), it should be understood that other devices may be configured to perform a method similar to that of FIG. 25.

[0235] In this example, video encoder 200 initially predicts the current block (400). For example, video encoder 200 may form a prediction block for the current block. Video

encoder 200 may then calculate a residual block for the current block (402). To calculate the residual block, video encoder 200 may calculate a difference between the original, unencoded block and the prediction block for the current block. Video encoder 200 may then transform the residual block and quantize transform coefficients of the residual block (404). Next, video encoder 200 may scan the quantized transform coefficients of the residual block (406). During the scan, or following the scan, video encoder 200 may entropy encode the transform coefficients (408). For example, video encoder 200 may encode the transform coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy encoded data of the block (410).

[0236] FIG. 26 is a flowchart illustrating an example method for decoding a current block of video data in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 24, it should be understood that other devices may be configured to perform a method similar to that of FIG. 26.

[0237] Video decoder 300 may receive entropy encoded data for the current block, such as entropy encoded prediction information and entropy encoded data for transform coefficients of a residual block corresponding to the current block (420). Video decoder 300 may entropy decode the entropy encoded data to determine prediction information for the current block and to reproduce transform coefficients of the residual block (422). Video decoder 300 may predict the current block (424), e.g., using an intra- or inter-prediction mode as indicated by the prediction information for the current block, to calculate a prediction block for the current block. Video decoder 300 may then inverse scan the reproduced transform coefficients (426), to create a block of quantized transform coefficients. Video decoder 300 may then inverse quantize the transform coefficients and apply an inverse transform to the transform coefficients to produce a residual block (428). Video decoder 300 may ultimately decode the current block by combining the prediction block and the residual block (430).

[0238] The following numbered clauses illustrate one or more aspects of the devices and techniques described in this disclosure.

[0239] Clause 1A. A method of coding video data, the method comprising: receiving a picture of video data; reconstructing the picture of video data; and performing a neural network (NN)-based filter process on the reconstructed picture of video data, wherein the NN-based filter process comprises performing a Canonical Polyadic (CP) decomposition with separable convolution.

[0240] Clause 2A. The method of clause 1A, further comprising refraining from performing a multiscale feature extraction in a residue block of the picture of video data.

[0241] Clause 3A. The method of clause 1A or clause 2A, further removing one or more activation layers in a head block.

[0242] Clause 4A. The method of any of clauses 1A-3A, wherein a group of inputs does not comprise an activation PReLU.

[0243] Clause 5A. The method of clause 4A, wherein the group of inputs comprises ReLU but does not comprise action PReLU.

[0244] Clause 6A. The method of any of clauses 1A-4A, wherein the group of inputs does not comprise any activations for head blocks.

[0245] Clause 7A. The method of any of clauses 1A-6A, further comprising refraining from applying a multiscale branch to backbone residue blocks.

[0246] Clause 8A. The method of clause 7A, further comprising applying the multiscale branch to at least one head block.

[0247] Clause 9A. The method of any of clauses 1A-8A, wherein the CP decomposition comprises: at least one 1x1 convolution; and a 3x1 separable convolutions in a horizontal dimension and a vertical dimension.

[0248] Clause 10A. The method of clause 9A, wherein the at least one 1x1 convolution comprises: a first 1x1 convolution occurring prior to the 3x1 separable convolutions; and a second 1x1 convolution occurring after the 3x1 separable convolutions.

[0249] Clause 11A. The method of clause 10A, wherein the second 1x1 convolution combines extracted features from previous layers.

[0250] Clause 12A. The method of any of clauses 1A-11A, wherein performing the CP decomposition with separable convolution comprises performing the CP decomposition to backbone and tail blocks.

[0251] Clause 13A. The method of any of clauses 1A-12A, wherein performing the CP decomposition with separable convolution comprises performing the CP decomposition to all blocks.

[0252] Clause 14A. The method of any of clauses 1A-13A, wherein coding comprises decoding.

[0253] Clause 15A. The method of any of clauses 1A-14A, wherein coding comprises encoding.

[0254] Clause 16A. A device for coding video data, the device comprising one or more means for performing the method of any of clauses 1A-15A.

[0255] Clause 17A. The device of clause 16A, wherein the one or more means comprise one or more processors implemented in circuitry.

[0256] Clause 18A. The device of any of clauses 16A and 17A, further comprising a memory to store the video data.

[0257] Clause 19A. The device of any of clauses 16A-18A, further comprising a display configured to display decoded video data.

[0258] Clause 20A. The device of any of clauses 16A-19A, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0259] Clause 21A. The device of any of clauses 16A-20A, wherein the device comprises a video decoder.

[0260] Clause 22A. The device of any of clauses 16A-21A, wherein the device comprises a video encoder.

[0261] Clause 23A. A computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to perform the method of any of clauses 1A-15A.

[0262] Clause 1B. A method of encoding or decoding video data, the method comprising: receiving data for reconstructing a picture of the video data; reconstructing the picture of the video data; and applying a neural network (NN)-based filter to the reconstructed picture of the video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0263] Clause 2B. The method of clause 1B, wherein the CP decomposition comprises: at least one 1x1 convolution; a convolution in a horizontal dimension; and a convolution in a vertical dimension.

[0264] Clause 3B. The method of clause 2B, wherein the convolution in the horizontal dimension comprises one of a 1x3 convolution or a 3x1 convolution, and wherein the convolution in the vertical dimension comprises another of the 1x3 convolution or the 3x1 convolution.

[0265] Clause 4B. The method of clause 3B, wherein the at least one 1x1 convolution comprises: a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension; and a second 1x1 convolution

occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension.

[0266] Clause 5B. The method of clause 4B, wherein the second 1x1 convolution combines extracted features from previous layers.

[0267] Clause 6B. The method of any of clauses 1B-5B, wherein the one or more backbone blocks and the tail block each comprise a corresponding CP decomposition with separable convolution.

[0268] Clause 7B. The method of any of clauses 1B-6B, wherein the head block, the transition block, the one or more backbone blocks, and the tail block each comprise a corresponding CP decomposition with separable convolution.

[0269] Clause 8B. The method of any of clauses 1B-7B, wherein the one or more backbone blocks do not include a multiscale branch.

[0270] Clause 9B. The method of any of clauses 1B-8B, wherein the head block does not include a corresponding parametric rectified linear unit activation (PReLU) for at least one input to the head block.

[0271] Clause 10B. The method of clause 9B, wherein the head block does not include a corresponding activation layer for at least one input to the head block.

[0272] Clause 11B. The method of clause 10B, wherein the head block does not include any activation layers.

[0273] Clause 12B. The method of any of clauses 1B-10B, wherein the head block includes at least one corresponding rectified linear unit activation (ReLU) for at least one input to the head block.

[0274] Clause 13B. The method of any of clauses 1B-12B, wherein the head block comprises at least one multiscale branch.

[0275] Clause 14B. The method of any of clauses 1B-13B, wherein the one or more backbone blocks each comprise 24 channels.

[0276] Clause 15B. A device for at least one of encoding or decoding video data, the device comprising: one or more memories configured to store the video data; and one or more processors, the one or more processors configured to: receive data for reconstructing a picture of the video data; reconstruct the picture of the video data; and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the

transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0277] Clause 16B. The device of clause 15B, wherein the CP decomposition comprises: at least one 1x1 convolution; a convolution in a horizontal dimension; and a convolution in a vertical dimension.

[0278] Clause 17B. The device of clause 16B, wherein the convolution in the horizontal dimension comprises one of a 1x3 convolution or a 3x1 convolution, and wherein the convolution in the vertical dimension comprises another of the 1x3 convolution or the 3x1 convolution.

[0279] Clause 18B. The device of clause 17B, wherein the at least one 1x1 convolution comprises: a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension; and a second 1x1 convolution occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension.

[0280] Clause 19B. The device of clause 18B, wherein the second 1x1 convolution combines extracted features from previous layers.

[0281] Clause 20B. The device of any of clauses 15B-19B, wherein the one or more backbone blocks and the tail block each comprise a corresponding CP decomposition with separable convolution.

[0282] Clause 21B. The device of any of clauses 15B-20B, wherein the head block, the transition block, the one or more backbone blocks, and the tail block each comprise a corresponding CP decomposition with separable convolution.

[0283] Clause 22B. The device of any of clauses 15B-21B, wherein the one or more backbone blocks do not include a multiscale branch.

[0284] Clause 23B. The device of any of clauses 15B-22B, wherein the head block does not include a corresponding parametric rectified linear unit activation (PReLU) for at least one input to the head block.

[0285] Clause 24B. The device of clause 23B, wherein the head block does not include a corresponding activation layer for at least one input to the head block.

[0286] Clause 25B. The device of clause 24B, wherein the head block does not include any activation layers.

[0287] Clause 26B. The device of any of clauses 15B-24B, wherein the head block includes at least one corresponding rectified linear unit activation (ReLU) for at least one input to the head block.

[0288] Clause 27B. The device of any of clauses 15B-26B, wherein the head block comprises at least one multiscale branch.

[0289] Clause 28B. The device of any of clauses 15B-27B, wherein the one or more backbone blocks each comprise 24 channels.

[0290] Clause 29B. The device of any of clauses 15B-28B, further comprising a display configured to display decoded video data.

[0291] Clause 30B. The device of any of clauses 15B-29B, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0292] Clause 31B. A device for at least one of encoding or decoding video data, the device comprising: means for receiving data for reconstructing a picture of the video data; means for reconstructing the picture of the video data; and means for applying a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0293] Clause 32B. Non-transitory storage media comprising instructions, that, when executed, cause one or more processors to: receive data for reconstructing a picture of video data; reconstruct the picture of the video data; and apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

[0294] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially.

[0295] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on

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

[0296] By way of example, and not limitation, such computer-readable storage media may include one or more of RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0297] Instructions may be executed by one or more processors, such as one or more DSPs, general purpose microprocessors, ASICs, FPGAs, or other equivalent integrated or discrete logic circuitry. Accordingly, the terms “processor” and “processing circuitry,” as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or

software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0298] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0299] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of encoding or decoding video data, the method comprising:
receiving data for reconstructing a picture of the video data;
reconstructing the picture of the video data; and
applying a neural network (NN)-based filter to the reconstructed picture of the video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.
2. The method of claim 1, wherein the CP decomposition comprises:
at least one 1x1 convolution;
a convolution in a horizontal dimension; and
a convolution in a vertical dimension.
3. The method of claim 2, wherein the convolution in the horizontal dimension comprises one of a 1x3 convolution or a 3x1 convolution, and wherein the convolution in the vertical dimension comprises another of the 1x3 convolution or the 3x1 convolution.
4. The method of claim 3, wherein the at least one 1x1 convolution comprises:
a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension; and
a second 1x1 convolution occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension.
5. The method of claim 4, wherein the second 1x1 convolution combines extracted features from previous layers.
6. The method of claim 1, wherein the one or more backbone blocks and the tail block each comprise a corresponding CP decomposition with separable convolution.

7. The method of claim 1, wherein the head block, the transition block, the one or more backbone blocks, and the tail block each comprise a corresponding CP decomposition with separable convolution.
8. The method of claim 1, wherein the one or more backbone blocks do not include a multiscale branch.
9. The method of claim 1, wherein the head block does not include a corresponding parametric rectified linear unit activation (PReLU) for at least one input to the head block.
10. The method of claim 9, wherein the head block does not include a corresponding activation layer for at least one input to the head block.
11. The method of claim 10, wherein the head block does not include any activation layers.
12. The method of claim 1, wherein the head block includes at least one corresponding rectified linear unit activation (ReLU) for at least one input to the head block.
13. The method of claim 1, wherein the head block comprises at least one multiscale branch.
14. The method of claim 1, wherein the one or more backbone blocks each comprise 24 channels.
15. A device for at least one of encoding or decoding video data, the device comprising:
 - one or more memories configured to store the video data; and
 - one or more processors, the one or more processors configured to:
 - receive data for reconstructing a picture of the video data;
 - reconstruct the picture of the video data; and

apply a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.

16. The device of claim 15, wherein the CP decomposition comprises:
 - at least one 1x1 convolution;
 - a convolution in a horizontal dimension; and
 - a convolution in a vertical dimension.
17. The device of claim 16, wherein the convolution in the horizontal dimension comprises one of a 1x3 convolution or a 3x1 convolution, and wherein the convolution in the vertical dimension comprises another of the 1x3 convolution or the 3x1 convolution.
18. The device of claim 17, wherein the at least one 1x1 convolution comprises:
 - a first 1x1 convolution occurring prior to the convolution in the horizontal dimension and the convolution in the vertical dimension; and
 - a second 1x1 convolution occurring after the convolution in the horizontal dimension and the convolution in the vertical dimension.
19. The device of claim 18, wherein the second 1x1 convolution combines extracted features from previous layers.
20. The device of claim 15, wherein the one or more backbone blocks and the tail block each comprise a corresponding CP decomposition with separable convolution.
21. The device of claim 15, wherein the head block, the transition block, the one or more backbone blocks, and the tail block each comprise a corresponding CP decomposition with separable convolution.

22. The device of claim 15, wherein the one or more backbone blocks do not include a multiscale branch.

23. The device of claim 15, wherein the head block does not include a corresponding parametric rectified linear unit activation (PReLU) for at least one input to the head block.

24. The device of claim 23, wherein the head block does not include a corresponding activation layer for at least one input to the head block.

25. The device of claim 24, wherein the head block does not include any activation layers.

26. The device of claim 15, wherein the head block includes at least one corresponding rectified linear unit activation (ReLU) for at least one input to the head block.

27. The device of claim 15, wherein the head block comprises at least one multiscale branch.

28. The device of claim 15, wherein the one or more backbone blocks each comprise 24 channels.

29. The device of claim 15, further comprising a display configured to display decoded video data.

30. The device of claim 15, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

31. A device for at least one of encoding or decoding video data, the device comprising:
- means for receiving data for reconstructing a picture of the video data;
 - means for reconstructing the picture of the video data; and
 - means for applying a neural network (NN)-based filter to the reconstructed picture of video data, wherein the NN-based filter comprises a unified filter comprising a head block, a transition block, one or more backbone blocks, and a tail block, and wherein at least one of the head block, the transition block, the one or more backbone blocks, or the tail block comprises a Canonical Polyadic (CP) decomposition with separable convolution.