

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

Published:

- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

**OPTIMIZATION TECHNIQUES FOR LOOP FILTERING USING NEURAL
NETWORKS**

CROSS-REFERENCE TO RELATED APPLICATIONS

- 5 [0001] This patent application claims the benefit of priority from Indian Provisional Patent Application Ser. No. 202311066258, filed on 3 Oct. 2023, Indian Provisional Patent Application Ser. No. 202311088224, filed on 22 Dec. 2023, and Indian Provisional Patent Application Ser. No. 202411050190, filed on 1 July 2024.

10 TECHNOLOGY

[0002] The present document relates generally to images. More particularly, embodiments of the present invention relate to optimization techniques for filtering images using neural networks.

15 BACKGROUND

[0003] In 2020, the MPEG group in the International Standardization Organization (ISO), jointly with the International Telecommunications Union (ITU), released the first version of the Versatile Video Coding standard (VVC), also known as H.266 (Ref. [6]). More recently, the same joint group (JVET) and experts in still-image compression (JPEG) have started
20 working on the development of the next generation of coding standards that will provide improved coding performance over existing image and video coding technologies. As part of this investigation, coding techniques based on artificial intelligence and deep learning are also examined. As used herein the term “deep learning” refers to neural networks having at least three layers, and preferably more than three layers.

25 [0004] As appreciated by the inventors here, improved techniques for the coding of images and video based on neural networks are described herein.

[0005] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this
30 section qualify as prior art merely by virtue of their inclusion in this section. Similarly, issues identified with respect to one or more approaches should not assume to have been recognized in any prior art on the basis of this section, unless otherwise indicated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] An embodiment of the present invention is illustrated by way of example, and not in way by limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0007] FIG. 1A depicts a network architecture with split luma-chroma architecture and CP-decomposition according to prior art;

[0008] FIG. 1B depicts a network architecture with fusion of convolution layer according to prior art;

[0009] FIG. 2A depicts a network architecture for a unified low-complexity loop-filter design according to prior art;

[00010] FIG. 2B depicts a network architecture for a unified low-complexity loop-filter design according to an embodiment of this invention;

[00011] FIGs 3A-3C depicts an example design process for a fuse-skip architecture according to an embodiment of this invention;

[00012] FIG. 4 depicts an example of window processing to derive a fuse-skip architecture according to an embodiment of this invention;

[00013] FIG. 5 depicts an example process to optimize the translation of a floating-point neural-network model to a fixed-point representation according to an embodiment of this invention;

[00014] FIG. 6A-FIG. 6G depict examples of reducing computational complexity according to embodiments of this invention;

[00015] FIG. 7 depicts a network architecture for a unified low-complexity loop-filter design according to an embodiment of this invention; and

[00016] FIG. 8A and FIG. 8B depict example backbone neural networks for improved training according to embodiments of this invention.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[00017] Example embodiments for loop filtering using neural networks in image and video coding are described herein. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the various embodiments of present invention. It will be apparent, however, that the various embodiments of the present invention may be practiced without these specific details. In other instances, well-known structures and devices are not described in exhaustive detail, in order to avoid unnecessarily occluding, obscuring, or obfuscating embodiments of the present invention.

SUMMARY

[00018] Example embodiments described herein relate to image and video coding using neural networks using optimization methods to reduce computational complexity. In an embodiment, given two consecutive (and neighboring) neural network (NN) blocks, each one with each own skip connection and two or more convolutional neural network blocks between the skip start and skip end of the skip connection, wherein the start or end point of the skip connection between two neural network layers in each NN block prohibits fusion of identical neighbor convolutional blocks, a sliding window method allows to move the skip start and skip end positions into new positions, thus allowing fusion of the neighboring identical neighboring blocks and reducing computational cost.

[00019] In a second embodiment, an iterative method to efficiently quantize a trained NN from a floating-point implementation to a fixed-point implementation is presented.

[00020] In other embodiments, additional methods to reduce operational complexity and improve training are also presented.

EXAMPLE CODING MODEL USING DEEP LEARNING

[00021] Deep learning-based image and video compression approaches are increasingly popular, and it is an area of active research. Current research in neural-networks (NN) based coding can be divided in two general frameworks: a “hybrid” neural network based framework, which simply replaces one or more existing coding or decoding modules with their corresponding neural-network-based implementation, where each NN module is trained and optimized on its own, and an “end-to-end” neural network, where training and optimizing is done on the whole network. The proposed neural network-based filters are applicable to either architecture. The term YUV420 denotes a luma-chroma color space, where chroma is sub-sampled by two in both the horizontal and vertical dimensions, such as YCbCr 4:2:0, and the like. While examples refer to a neural network-based loop filter (NNLF), the techniques are applicable to a variety of other NN-based filters which remove noise artifacts or improve image quality, such as NN post filtering, super-resolution filtering, and the like. Embodiments presented herein offer improved coding efficiency at a reduced computational cost.

[00022] NNLf technologies have been actively studied in the Joint Video Experts Team (JVET) as part of exploratory experiments 1 (EE1). For example, Ref. [4], which describes a low complexity loop filter, has been adopted in JVET’s NN code base version NNVC-5.0 as

the low operation point (LOP) anchor. Meanwhile, a unified LOP filter has been proposed in the July 2023 JVET meeting, which also incorporated a split luma-chroma design, first proposed in Ref. [4]. Based on a proposed model of the unified LOP architecture, example embodiments of several skip-connection designs in a general form are now proposed. Given
5 that some of the proposed skip connection designs could enable the fusion of adjacent 1x1 convolutional layers, further reductions in complexity are also possible. Furthermore, an adaptive quantization method is also proposed to efficiently quantize the model weights and parameters after training in a cost-efficient integer-point implementation.

10 Split Luma Chroma NNLF with Skip Connections

[00023] Ref.[4] proposed a neural network-based loop filter method that incorporated CP decomposition. In CP-decomposition (CP comes from a CANDECOMP/PARAFAC model), a 4D convolution kernel tensor is decomposed into a sequence of four convolutional layers
15 with small kernels. The first convolution layer is a pointwise convolution, the second and third layers are spatial convolutions in X and Y directions, and the fourth convolution is again a pointwise convolution in the channel dimension.

[00024] In Ref. [4], the key idea is to first decompose the 3x3 regular convolution into a 1x1 point-wise convolution, a 3x1 depth-wise separable convolution, a 1x3 depth-wise separable
20 convolution, and a 1x1 point-wise convolution using CP decomposition. Next, the channels are split into a luma path and a chroma path, where the luma path usually has a larger number of channels than the chroma, as the luma component needs more parameters and weights to process. Finally, as shown in FIG. 1B, adjacent 1x1 convolutions after CP decomposition are fused (Ref. [4]).

[00025] Compared to earlier baseline models, the complexity in kMac/pixel is reduced by about 50% while the performance remains almost the same. The term kMac/pixel stands for “thousand multiply-accumulation operations per pixel.” The version without split luma and chroma, i.e., the one with CP decomposition and fusion, as depicted in FIG. 1B (Ref.[4]), has
25 been adopted as the NNVC-5.0 reference software anchor in JVET.

[00026] Using a larger kernel size for convolutional neural networks usually provides larger receptive fields and improved performance. However, for regular convolution with kernel size $M \times M$, as M increases the computation complexity increases proportionally to M^2 . Therefore, stacking multiple layers with smaller kernel size, such as 3x3, is the common strategy to
30 achieve a good performance with complexity constraint. In previous designs, separable

convolutions with kernel size 1×3 and 3×1 are used to replace a regular 3×3 convolution. For separable convolutional layers with kernel size $1 \times M$ or $M \times 1$, the convolution operation of each layer is only performed in one dimension. Thus, when setting the kernel size to M , the complexity increases proportionally to M instead of M^2 . Considering this, in an embodiment a kernel of size M , where M is larger than 3, is used in the $1 \times M$ or $M \times 1$ kernels to improve the performance of the neural-networks. Larger kernel sizes may also be used in separable $M \times M$ convolutions. An $1 \times M$ and an $M \times 1$ kernel may be fused into a separable $M \times M$ kernel, while reducing the number of layers. This would yield lower latency with marginal increase in computational complexity.

[00027] Consider a neural network consisting of n backbone blocks, where each backbone block i contains $1 \times M_i$ and $M_i \times 1$ convolutions, where, without limitation, $M_i \in [3, 9]$, for $i = 1, 2, \dots, n$. For example, for a task that needs a larger receptive field but with a computation complexity constraint, one may use a larger kernel size, say $M_i = 5$, for any of the n backbone blocks and/or any other input, hidden, or output convolutional layers in the neural-network, while reducing the number of backbone blocks (n) or the number of any other layers to reduce complexity. The value of M_i can also be set differently for different backbone blocks and any other convolutional layers in the neural network. For example, the first backbone blocks can use a kernel value of 5 to process the input better while the last ones can use smaller value, say 3, to meet the complexity constraints while reducing the number of layers to further reduce overall complexity.

[00028] As depicted in FIG. 1B, convolution layers 105 and 115 can be fused to generate convolution layer 130, and convolution layers 125 and 110 can be fused to generate convolution layer 140. This fusion helps in reducing the number of layers in the network, which can help in lowering latency of the network and memory access overhead and can take advantage of optimized hardware/software implementations. This fusion is possible because there are no nonlinear activations present between the layers being fused and the fused block is mathematically equivalent to the layers being fused.

[00029] During the July 2023 meeting of JVET, a unified Low Operation Point (LOP) NNLF was proposed jointly by several proponents (Ref. [2]). A key element of that proposal is a split luma-chroma architecture (Ref. [4-5]). FIG. 2A depicts a block diagram of the proposed architecture. As depicted in FIG. 2A, compared to the architecture in Ref. [4], the inputs, headblocks and tailblocks in the common path are different, which actually references the unified High Operation Point (HOP) NNLF design in Ref. [3], while the number of

parameters has been modified so that the kMac/pixel computational load is below 20. However, the basic backbone block in this design is still very similar to the architecture in FIG 1A. The major difference is that in FIG. 2A, a regular skip connection (207), which is the same as the one in the HOP backbone, is added and because the end point of the skip connection and the start point of the skip connection of the next backbone block are between two 1x1 convolution layers, the fusion of these two is not possible anymore.

[00030] To address this issue, embodiments presented herein propose to have alternative skip connection designs that could enable the fusion of the two adjacent identical (e.g., 1x1) convolutions. In other words, as long as the start and end point of the skip connection don't fall between this two adjacent 1x1 convolutions, the fusion of the two is hence possible and should be performed to further reduce complexity.

[00031] One example embodiment of the backbone with fusion-skip connection is depicted in FIGs 3A-FIG.3C. The idea is that, given a series of backbone blocks (205) one can move the start point (220) of the original skip connection to the right by one layer, i.e., the start point of the new skip connection (320) is right before the non-linear activation (225-1) while the end point is before the non-linear activation (225-2) of the next backbone. Then the last two 1x1 convolution networks (210 and 215) could be fused into one 1x1 convolution (305) as there is no more skip connection (e.g., 320) between the pair. This new design can also be interpreted as a new backbone module architecture/order of layers (310), as depicted in FIG. 3C. Using the new backbone design (310) of FIG. 3C, due to fusion, in an embodiment, the complexity of the modified FIG. 2A design is reduced from about 17.0 kMac/pixel to about 16.5 kMac/pixel. For example, a 1920 x 1080 frame has 2.07 M pixels, so this method can save about 1.03 Giga multiply-accumulate operations per frame. In an embodiment, in the new backbones (e.g., 310), the positions of the PReLU (225-1) and its subsequent Conv 1x1 can be swapped, bringing the Conv 1 x 1 block in the top of the backbone. This has no effect on the computational complexity of the new backbone.

[00032] As depicted in FIG. 3B, the proposed shift of a skip connection may leave certain layers in the first and last backbone layers of a chain of backbones as "leftovers" or "dangling," respectively. For example, note that the initial 1x1 convolutional layer (230) before s_0 of the first new backbone layer (e.g., 230) and some remaining layers (e.g., the PReLU, parametric rectified linear unit, (225) and all subsequent convolutional blocks until the end) of the last backbone layer (e.g., the last backbones, N_Y or N_C in FIG. 2A) would be left dangling. In an embodiment, one may leave the dangling layers as is and not add any skip connections to them. Then, the number of backbone layers in the new design of FIG. 2A will be N_Y-1 and N_C-1 .

Alternatively, a single 1x1 convolution layer may now be added before the first backbone layer and after the last backbone layer in the new design, to match the channel dimensionality of the preceding head-end layers and the subsequent tail-end layers respectively. In FIG. 2A, the proposed changes should be made in both the Luma and the Chroma paths, so that the number of backbone layers in the new design remain as N_Y and N_C .

[00033] The process described in FIGs 3A-3C can be generalized using a sliding window approach. As shown in FIG. 4, the original backbone blocks of FIG. 3A are depicted in dotted rectangles (205-1, 205-2). For each backbone block, there are six layers with corresponding layer locations denoted as “ $k-i$ ”, $k=0$, and $i=0$ to 5 for backbone 205-1 and $k=1$, $i=0$ to 5 for backbone 205-2. For a cascaded NN block network, subsequent backbone blocks could follow with $k=2, 3, \dots, K-1$, where K denotes the number of total backbone blocks in the network. For example, in the first backbone block, the input is denoted as 0-0, which is the input point before the first 1x1 convolutional layer. For a skip connection in a backbone block, the start point is denoted as s while the end point is denoted as e . At point s , the data is simply extracted.

At point e , the data from location s is added to the actual location of e using element-wise adding. If point s_0 starts at location 0-0, then point e_0 ends at location 1-0, which covers 6 layers. Correspondingly, the start-point s in the next backbone (205-2), s_1 , always has the same location right after the previous end point e_0 , i.e., s_1 takes the data of previous layer's output (e.g., e_0) plus the data from s_0 using element-wise adding. When the location of s_0 is at 0-0, the whole architecture is the same as the one in FIG. 2A. In such a scenario, e_0 and s_1 are between two adjacent 1x1 convolution networks so that the two cannot be fused.

[00034] If one moves the start point s_0 from 0-0 to 0-1, (and the start point s_1 from 1-0 to 1-1) the architecture is the same as the one in FIG. 3C. Since there is no start or end point between the two 1x1 convolutions between points 0-5 and 1-1, the two can be fused into one single 1x1 convolution. Similarly, one can move start point s_0 to the right at 0-2, 0-3, 0-4 and 0-5, resulting in alternative architectures with different skip connections. All such designs allow the fusion of the two adjacent 1x1 convolutions between points 0-5 and 1-1 and reduce the number of layers in one backbone from 6 to 5. Meanwhile, the actual backbone's range (e.g., denoted by dotted rectangle 205-1) should also be moved accordingly (e.g., see 310). For subsequent backbone blocks, s_j is always right after e_{j-1} , therefore, when s_1 and e_0 , i.e., the first skip-end connection is moved, all latter backbone skip connections should be moved accordingly.

[00035] Examples described so far assume that neighboring blocks to be considered are identical and end point e_j and start point s_{j+1} always coincide. They also assume that in a chain of K blocks all start and end points may need to be shifted. This may not be always the case.

For example, in other embodiments, the front-end and/or the back-end block in series of consecutive blocks may be slightly different. Thus, for example, one may retain s_0 and e_0 as is, in their original position, and only adjust the position of s_j and e_j in subsequent blocks. Regardless of the specifics, a person skilled in the art would appreciate how to apply the proposed windowing method to enable fusion between neighboring identical networks or layers (e.g., convolutional networks) that are separated by a skip connection.

[00036] As discussed earlier, in other embodiments, in the architectures depicted in FIG. 2B, and FIG. 3A-3C, to improve performance and reduce complexity, one may use kernels larger than 3×1 or 1×3 , but reduce the number of layers. In an embodiment, when using a kernel size of 5, and reducing the number of backbone blocks in the luma path (N_Y) from 14 to 13, then the overall complexity is reduced from about 16.9 kMac/pixel to 16.3 kMac/pixel while the performance slightly improves.

Adaptive quantization of an NNLF model

[00037] This section describes embodiments to quantize the floating-point convolution weights and bias layers of a neural network to realize an efficient fixed-point neural network without affecting its performance in terms of accuracy of the output.

[00038] Proposed embodiments of this invention are aimed at realizing the integer implementation of NNLF. Without limitation, and as an example, the integer implementation and verification are done using SADL (Small Ad hoc Deep-Learning Library), which is a light-weight library to perform neural-network inference in pure C++. Without limitation, and as an example, the performance of fixed-point implementations will be measured based on how it performs compared to its floating-point implementation for Random Access, Main 10, for the JVET “Class D” set of test sequences due to its low resolution and therefore fast inference.

[00039] The overflow is detected with the help of inference for the selected test contents. The SADL library already has the utility to detect and print overflows, hence the same utility is used during the process. An example embodiment of the process is depicted in FIG. 5. As depicted in FIG. 5, during an initialization step (505)

- a. Set initial quantization factors (Q factors) for all layers with a constant value. As an example, for a 16-bit integer model, Q factors may be set to Q11, which indicates that 5 bits are used for sign and integer part, and 11 bits are used for the fractional part, for a total of 16 bits.

b. In the inference software, enable the existing SADL overflow prints utility. This utility prints the total number of overflows out of the total number of operations for each layer. The prints are redirected to the encoder logs.

c. Compute the performance of the trained floating-point model.

5 [00040] Next (510), the floating-point model is converted to a fixed-point model (e.g., 16-bit) by quantizing the weights and biases with the set Q factor. This is used as the input model for the next step.

[00041] Next (515), inference is conducted for selected test sequences with the input model. There is no restriction for the input sequences that can be used. For example, without limitation,
10 one may use the JVET Class D test sequence, and the inference can be conducted for the Random Access Main 10 configuration. The overflow prints are dumped to the encoder logs by the SADL utility during this process.

[00042] In step 520, the performance of the new fixed-point model is compared with the floating-point model. As an example, the performance may be measured based on a bit-rate vs
15 distortion (BD) rate difference (e.g., bit rate vs peak signal to noise ratio (PSNR)). If the difference between them is below a performance threshold (522) (the BD-rate threshold) the process is terminated, and the current fixed-point model is declared to be the optimal model (535), otherwise the process continues. The performance threshold is configured by the user. For example, in an embodiment, a value of 0.10% was used.

20 [00043] In step 525, one records (e.g., using a Python script) the overflow percentage for all the layers in the model, for all the encoded sequences.

[00044] Next (530), the integer model is adjusted as follows: the quantization factors (Q factors) are reduced by a set value T (e.g., $T=1$) for the layers that have overflows higher than a user-defined overflow threshold 527 (e.g., 0.01%). The layers which have no overflows or
25 overflow less than the threshold will use the previously set Q factors. The process continues from step 510, until it terminates in step 535.

Further Complexity Reduction Techniques

[00045] FIG. 2B depicts a network architecture for a unified low-complexity loop-filter
30 design according to an example embodiment of this invention using the complexity reduction techniques proposed earlier. For example, compared to backbone network 205, backbone network 205B (identical to backbone block 310) does not have the 1x1 CxC1 convolution block before the first PReLU, due to fusion with the last 1x1 convolution block. Due to this fusion, compared to FIG. 2A, the architecture in FIG. 2B features also new 1x1 convolution networks

before the first backbone block and after the last back-bone block in each of the split luma and chroma paths.

[00046] Note that from Ref. [2], in FIG. 2A and FIG. 2B: in the headblock (250) $\mathbf{d} = [d_1, d_2, d_3, d_4, d_5, d_6] = [12, 8, 4, 2, 2, 24]$, and in the backbone (205, 205B), without limitation, example block parameters may be as in Table 1.

Table 1. Example backbone neural-network parameters

Parameter	Luma	Chroma
N	$N_Y = 14$	$N_C = 4$
C	$C_Y = 16$	$C_{UV} = 16$
C1	$C1_Y = 64$	$C1_{UV} = 32$
C21	$C21_Y = 16$	$C21_{UV} = 16$

[00047] As discussed earlier, video coding architectures based on neural networks, such as the LOP NNFL depicted in FIG. 2A and FIG. 2B, provide increased coding efficiency, but at a much higher computational cost. Using the LOP filter in FIG. 2B as an example and without limitation, in the next sections a variety of complexity reduction techniques will be proposed. These techniques are applicable to all neural-network-based architectures with components identical or similar to those used by the LOP filter.

1x1 fusion down-sampler

[00048] In FIG. 2B, in the fusion and transition blocks 240, in an embodiment, the 3x3 convolutional layer with down sampler (242) is removed along with its activation layer, PReLU 244, to reduce complexity. The down-sampling operation that is present is being shifted to the preceding 1x1 convolution layer (246). The combined operation (605), which includes a new 1x1 convolutional block with down sampler (607) and the original PReLU (244), as depicted in FIG. 6A, can be used in place of the fusion and transition blocks (240) and reduces computational complexity by about 17%. In an alternative embodiment, one could retain the 3x3 convolution layer itself, but move the down-sampling to the 1x1 convolutional layer 246, but this may result to higher computational complexity than the one achieved with the proposed simplification without the 3x3 convolution layer.

Headblock down-sampler

[00049] In FIG. 2B, the input 1x1 convolutional layers (250) that are in parallel and take as input: reconstructed (Rec.), predicted (Pred.), boundary strength (BS), QPbase, QPslice, and IPB inputs, are referred to as head-blocks. In an example embodiment, as depicted in FIG. 6B, the down-sampling operations in convolutional blocks 242 (or in 607), can be shifted to the headblock convolutional layers. The kernel size of all the convolutional layers of the new headblock (610) is set to 3 since the loss of information from the input is higher with the kernel size 1 compared to kernel size 3 during the down-sampling. Changing all the headblock kernel sizes to 3 results in negligible increase in complexity because the convolutional layers in the headblock (250) for the Pred and Rec inputs already use a kernel size of 3.

[00050] Optionally, the down-sampling operation can be moved to the headblock convolutional layers by retaining the original 3x3 convolutional layer (242) of the transition block (240), but without down-sampling. However, this can lead to higher complexity compared to removing the 3x3 transition block (242) and using instead the architecture in FIG. 6A. Block 605 is now replaced by block 615, where 1x1 convolutional layer 617 has no down-sampling operation.

3x3 down-sampler CPD

[00051] In an alternative embodiment, in FIG. 2B, in the fusion and transition blocks 240, the 3x3 convolutional layer with down sampler (242) may be decomposed using the canonical polyadic decomposition (CPD). As depicted in FIG. 6C, in block (620), this would replace the 3x3 down-sampler convolutional layer (242) with the sub-blocks in (622): a 1x1 convolutional layer (628), 1x3 and 3x1 separable convolutional layers, and a 1x1 convolutional layer (624). This reduces the complexity of the architecture by 6%. It is to be noted that this technique can be combined with both techniques described earlier by not doing the down-sampling operation in the 1x3 and 3x1 separable convolutional layers of 622, but moving them either to the convolution block 246, or in the headblock. Note also that a similar scheme can be used for $M \times M$ convolutional layers, for $M > 3$. That is, an $M \times M$ convolutional layer with down-sampler can be replaced with: a 1x1 convolutional layer, $1 \times M$ and $M \times 1$ separable convolutional layers, and a 1x1 convolutional layer.

3x3 down-sampler CPD with fusion

[00052] The CPD-based design (620) described in the previous section has two 1x1 convolutional layers separated by non-linear activation layer PReLU. The first 1x1 convolutional layer (246) is the layer present after the headblock (250). The second 1x1 layer

(628) is the layer that got created from the CPD of the 3x3 down-sampler layer. The PReLU layer (248) present between the two 1x1 convolutional layers is blocking the fusion of those two layers, a fusion that would help in reducing the complexity of the architecture. In an embodiment, the PReLU layer (248) present between the two adjacent 1x1 convolutional layers is moved before the first 1x1 convolutional layer (246), after the headblock, allowing the two 1x1 convolutional layers to be fused into single 1x1 convolutional layer. The architecture is shown in FIG. 6D. As depicted in FIG. 6D, in block (625), the 1x1 convolution layer 246 has been eliminated, since it is fused with 1x1 convolution layer 628, which is part of the 3x3 CPD layer (622). Block 625 can now replace block 240. This technique is also applicable to $M \times M$ down-sampler CPD blocks.

[00053] This operation further reduces the complexity by ~6% from the 3x3 down-sampler CPD architecture, and by ~11% from the original LOP architecture.

[00054] It is to be noted that similarly to the previous 3x3 (or $M \times M$) down-sampler CPD technique, this is independent of the first two techniques that were described earlier. Hence this technique can be combined with the first two techniques by not down-sampling in the 3x1 (or $M \times 1$) and 1x3 (or $1 \times M$) separable convolutional layers in CPD block 622.

Combining tail blocks into the backbone

[00055] In FIG. 2B, the layers present after the split luma-chroma backbone layers are referred to as tail blocks. From FIG. 2B, it can be observed that the convolutional layers present in the tail block before the PReLU (260) are similar to the convolutional layers present after the PReLU in the backbone (205B). A minor dimensionality difference exists between the 1x1 convolutional layer before the pReLU (260) in the tail block and the last 1x1 convolutional layer of the backbone. Utilizing the described similarity, in an embodiment, the convolutional layers in the tail block before the PReLU (260) are replaced with the backbone layer, thus increasing the total backbone layers now to N_Y+1 and N_C+1 . This will increase the complexity due to the additional PReLU being added from the converted backbone layer, and the increased input to the output/last 3x3 convolutional layers of the tail block, but this architecture will give the advantage of the skip connection which comes as a byproduct during the conversion of the tail block layers to a backbone block. The converted tail block with an integrated backbone is depicted in FIG. 6G. In another embodiment, one can still maintain the same number of backbones as before (e.g., N_Y , N_C), but this may result in some loss in performance. An example of the overall architecture with N_Y and N_C backbones is depicted in FIG 7.

[00056] It is to be noted that this technique is independent of all the other techniques described earlier. Hence this can be combined with other techniques. For example, block 240 can be replaced by either block 605, or the combination of blocks (610) and (615), or block (620), or block (625). Which brings up two more possible fusions of convolutional blocks to reduce operational complexity. When replacing block (240) with block (625), one can fuse the end 1x1 convolution (624) in CPD block 622 with the 1x1 convolutional blocks (710 and 712) before the first backbone blocks in the split luma and chroma paths. There are two possible fusion schemes:

- keep the 1x1 conv 624 in the CPD block, but remove the two 1x1 conv blocks (710 and 712); or
- remove the 1x1 conv 624 in the CPD block, and keep the two 1x1 conv blocks (710 and 712)

[00057] As depicted in FIG. 2B and FIG. 7, since backbone 705 starts with a PReLU, the last PReLU (626) of block (625) can now be removed to form the simplified block (630). As depicted in FIG. 6E, the last 1x1 convolutional layer (637) in the CPD (635) will be fused with the two 1x1 convolutional blocks (710 and 712) to form a single 1x1 convolutional block with d_6 inputs and $C_{1Y}+C_{1UV}$ outputs. The number of multiplications saved by removing convolution blocks 710 and 712 is $(C_Y * C_{1Y} + C_{UV} * C_{1UV})$. Considering that operations will be in the sub-sampled domain, one can show that after fusion there will be a change of $\{d_6 * (C_{1Y} - C_Y + C_{1UV} - C_{UV}) - (C_Y * C_{1Y} + C_{UV} * C_{1UV})\} / 4$ kMac/pixel operations. Using $d_6=24$ and the C and C1 backbone parameters in Table 1, this approach does not offer any computational reduction, but could offer an advantage for other values of d_6 , C, and C1.

[00058] In the second approach, as depicted in FIG. 6F, in block 640 (to replace block 625), in CPD 645, the end 1x1 convolution block is now removed, and the output is now directly from the separable convolution with down-sampling. The d_6 outputs of the 3x1 separable convolutions needs to be split into d_{6Y} and d_{6UV} ($d_6 = d_{6Y} + d_{6UV}$). Similarly, the 1x1 conv blocks 710 and 712 will use only d_{6Y} and d_{6UV} input channels. This approach will always reduce complexity. One can show that after fusion there will be a change of $\{(d_{6Y} * C_{1Y} + d_{6UV} * C_{1UV}) - d_6 * (C_Y + C_{UV}) - (C_Y * C_{1Y} + C_{UV} * C_{1UV})\} / 4$ kMac/pixel operations. By splitting $d_6 = 24$ into 16 d_{6Y} and 8 d_{6UV} channels, and using the same parameters as earlier (Table 1), this approach can save 0.256 kMacs per pixel.

[00059] A similar fusion approach could also be considered in the networks depicted in FIG. 6B. For example, in an embodiment, 1x1 conv block (617) could be moved to be after the PReLU (244), thus allowing it to be fused again with the 1x1 conv blocks 710 and 712.

5 Training improvements

[00060] As depicted in FIG. 8A, in an embodiment, the backbone (705) in FIG. 7 can be further optimized for training by adding a normalization connection (802). Let the input to the whole block be denoted as X , and let the output (804) after the five layers be $F(X)$. $F(X)$ is multiplied element-wise with the input X . Then, the $F(X) \cdot X$ product is added to the original
10 input X . Therefore, the final output Y of the new backbone becomes $Y = F(X) \cdot X + X$. Compared to the original design, designed to optimize learning the residual ($Y - X$), the new architecture is aimed at learning the normalized version of the residual. Thus, the new architecture should have better generalization. An additional advantage of this approach is that one can also add a variable α , such that $Y = \alpha \cdot F(X) \cdot X + X$. Variable α can be used for content-adaptive optimization.

[00061] To make training easier and more stable for the new backbone in FIG. 8A, in an embodiment, as depicted in FIG. 8B, one may add a sigmoid activation in the normalization connection. The original input X will go through the Sigmoid layer and it will be converted to a value larger than 0. Then, it is element-wise multiplied with the output after the five layers.

[00062] In an alternative embodiment, instead of using a sigmoid, in FIG. 8B one may use
20 a 1x1 separable convolution instead, or a combination of a 1x1 separable convolution followed by a nonlinear activation layer, such as an ReLU, a PReLU, a sigmoid, and the like (not shown), depending on the training of the network and its application.

References

25

Each one of the references listed herein is incorporated by reference in its entirety. The term JVET refers to the Joint Video Experts Team of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29.

30

- [1] J. N. Shingala, et al., "AHG11: Complexity reduction on neural-network loop filter," JVET-AA0080-v2, teleconference, July 2022.
- [2] D. Rusanovskyy, et al. "Unified LOP filter design, training procedure and filter usage," JVET-AE0281, Geneva, CH, July 2023.

[3] F. Galpin, et al., "AhG11 - EE1-0 High Operation Point model," JVET-AE0191-v6, Geneva, CH, July 2023.

[4] J.N. Shingala, et al., "Loop Filtering using neural networks," PCT Patent Application PCT/US2023/026238 (D22063WO01), filed on 26 June 2023.

5 [5] J.N. Shingala, et al., "EE1-4.1: Neural-network loop filters with further complexity reduction," JVET-AE0067, Geneva, CH, July 2023.

[6] *Versatile Video Coding*, Rec. ITU-T H.266, August 2020, ITU.

[7] T. Shao, et al., "EE1-related: Further complexity reduction on the joint EE1-0," JVET-AF0071, Hannover, Germany, Oct. 2023.

10

EXAMPLE COMPUTER SYSTEM IMPLEMENTATION

[00063] Embodiments of the present invention may be implemented with a computer system, systems configured in electronic circuitry and components, an integrated circuit (IC) device such as a microcontroller, a field programmable gate array (FPGA), or another configurable or
15 programmable logic device (PLD), a discrete time or digital signal processor (DSP), an application specific IC (ASIC), and/or apparatus that includes one or more of such systems, devices or components. The computer and/or IC may perform, control, or execute instructions relating to loop filtering using neural networks for image and video coding, such as those described herein. The computer and/or IC may compute any of a variety of parameters or
20 values that relate to loop filtering using neural networks for image and video coding described herein. The image and video embodiments may be implemented in hardware, software, firmware and various combinations thereof.

[00064] Certain implementations of the invention comprise computer processors which execute software instructions which cause the processors to perform a method of the invention.
25 For example, one or more processors in a display, an encoder, a set top box, a transcoder, or the like may implement methods related to loop filtering using neural networks for image and video coding as described above by executing software instructions in a program memory accessible to the processors. Embodiments of the invention may also be provided in the form of a program product. The program product may comprise any non-transitory and tangible
30 medium which carries a set of computer-readable signals comprising instructions which, when executed by a data processor, cause the data processor to execute a method of the invention. Program products according to the invention may be in any of a wide variety of non-transitory and tangible forms. The program product may comprise, for example, physical media such as magnetic data storage media including floppy diskettes, hard disk drives, optical data storage

media including CD ROMs, DVDs, electronic data storage media including ROMs, flash RAM, or the like. The computer-readable signals on the program product may optionally be compressed or encrypted.

[00065] Where a component (e.g. a software module, processor, assembly, device, circuit, etc.) is referred to above, unless otherwise indicated, reference to that component (including a reference to a "means") should be interpreted as including as equivalents of that component any component which performs the function of the described component (e.g., that is functionally equivalent), including components which are not structurally equivalent to the disclosed structure which performs the function in the illustrated example embodiments of the invention.

[00066] Various aspects of the present disclosure may be appreciated from the following Enumerated Example Embodiments (EEEs):

[00067] EEE 1. A method to optimize neural networks with cascaded block networks, the method comprising: accessing a first block network comprising: a first skip connection between a first start point and a first end point, and two or more first-stage convolutional networks between the first start point and the first end point; accessing a second block network subsequent to the first block network comprising: a second skip connection between a second start point and a second end point, wherein the second start point coincides with the first end point, and two or more second-stage convolutional networks between the second start point and the second end point; moving the first start point of the first skip connection to a new position in between the two or more first-stage convolutional networks but before the first end point; moving the first end point of the first skip connection and the second start point of the second skip connection to the same new position in between the two or more second-stage convolutional networks; fusing identical networks between the moved first start point and the moved first end point to generate at least one fused convolutional network; and generating an output block network comprising: the moved first start point, the moved end point, a skip connection between the moved start point and the moved end point; and at least the one fused convolutional network between the moved first start point and the moved first end point.

[00068] EEE 2. The method of EEE 1, wherein the two or more first-stage convolutional networks between the first start point and the first end point comprise: a first 1x1 convolution network (CN), followed by a first non-linear activation, followed by a second 1x1 CN, followed by a first 1x3 CN, followed by a first 3x1 CN, followed by a third 1x1 CN, and the two or more second-stage convolutional networks between the second start point and the second end point comprise: a fourth 1x1 convolution network (CN), followed by a second non-linear

activation, followed by a fifth 1x1 CN, followed by a second 1x3 CN, followed by a second 3x1 CN, followed by a sixth 1x1 CN.

[00069] EEE 3. The method of EEE 2, wherein the start point of the first skip connection is moved after the first 1x1 CN and before the first non-linear activation, the end point of the first skip connection is moved after the fourth 1x1 CN and before the second non-linear activation, and the third 1x1 CN is fused with the fourth 1x1 CN to generate a fused 1x1 CN.

[00070] EEE 4. The method of EEE 3, wherein the output block network comprises: the first non-linear activation, followed by the second 1x1 CN, followed by the first 1x3 CN, followed by the first 3x1 CN, and followed by the fused 1x1 CN.

[00071] EEE 5. A method to generate quantization factors to translate a floating-point neural-networks model to an integer-point neural-networks model, the method comprising: generating a floating-point performance metric using a floating-point model of a neural network (NN) and a set of test sequences; determining initial Q factors for weights and biases to generate a fixed-point model of the NN; contacting inference of the fixed-point model using the set of test sequences and computing a fixed-point performance metric of the fixed-point model; if the difference between the floating-point performance metric and the fixed-point performance metric is smaller than a performance threshold, then determining an output fixed-point model using Q factors generated up to this step, else: determining overflow values across all layers of the fixed-point model; generating updated Q factors, wherein generating updated Q factors comprises: for each overflowed layer, if its overflow is larger than an overflow threshold, then decreasing the Q factors in that layer by a fixed value, else leaving the Q factors in that layer unchanged; and returning to the inference step using the updated Q factors.

[00072] EEE 6. The method of EEE 5, wherein for a 16-bit fixed-point model the initial Q factors are set at Q11.

[00073] EEE 7. The method of EEE 6, wherein Q factors in an overflowed layer are decreased by 1.

[00074] EEE 8. An apparatus for video loop filtering using neural-network components, the apparatus comprising: a headblock (250) comprising a series of inputs, each input followed by a 1x1 or 3x3 convolutional network; a fusion and transition (FT) block (605) following the headblock; a series of one or more back-bone blocks (205B) following the fusion and transition block; and a tail block following the one or more back-bone blocks, wherein the fusion and transition block (605) comprises a 1x1 convolution block with down-sampling (607) followed by a first activation layer (PReLU).

[00075] EEE 9. The apparatus of EEE 8, wherein in the FT block (615), the 1x1 convolution

block with down-sampling (607) block is replaced with a 1 x 1 convolution block (617), and in the headblock (610) each convolutional network is replaced by a 3x3 convolutional network with down-sampling.

[00076] EEE 10. An apparatus for video loop filtering using neural-network components,
5 the apparatus comprising: a headblock (250) comprising a series of inputs, each input followed by a convolutional network; a fusion and transition (FT) block (620) following the headblock; a series of one or more back-bone blocks (205B) following the fusion and transition block; and a tail block following the one or more back-bone blocks, wherein the FT block (620) comprises:
10 a first 1x1 convolutional (CONV) block (246), followed by a first activation layer (PReLU) (248), followed by a 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling (622), followed by a second activation layer (PReLU), wherein the CPD block comprises: a second 1x1 CONV block (628), followed by a 1x3 separable CONV with down sampling, followed by a 3x1 separable CONV with down sampling, followed by a third 1x1 CONV block (624).

15 [00077] EEE 11. The apparatus of EEE 10, wherein the FT block (620) is replaced by a second FT block (625), the second FT block comprising: the first activation layer (PReLU) (248), followed by the 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling (622), followed by the second activation layer (626).

[00078] EEE 12. The apparatus of EEE 10, wherein the FT block (620) is replaced by a third
20 FT block (630), the third FT block comprising: the first activation layer (PReLU) (248), followed by a second 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling (635 or 645).

[00079] EEE 13. The apparatus of EEE 12, wherein the second 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling (645) comprises: the second 1x1 CONV
25 block (628), followed by a second 1x3 separable CONV with down sampling, followed by a second 3x1 separable CONV with down sampling.

[00080] EEE 14. An apparatus for video loop filtering using neural-network components, the apparatus comprising: a headblock (250) comprising a series of inputs, each input followed by a convolutional network; a fusion and transition (FT) block following the headblock; a series
30 of one or more back-bone blocks (705) following the fusion and transition block; and a tail block following the one or more back-bone blocks, wherein after the last back-bone block the tail block comprises: an activation layer (PReLU) (260), followed by a 3x3 convolutional layer.

[00081] EEE 15. The apparatus of EEE 14, wherein a back-bone comprises: an activation layer (PReLU), followed by a first 1x1 CONV block, followed by a 1x3 separable CONV,

followed by a 3x1 separable CONV, followed by a second 1x1 CONV block, followed by a scaler (802), wherein the scaler multiplies an input to the back-bone with the output (804) of the second 1x1 CONV block, followed by an adder, wherein the adder adds an input to the back-bone with the output of the scaler.

5 [00082] EEE 16. The apparatus of EEE 15, wherein the positions of the activation layer (PReLU) and the first 1x1 CONV block are swapped.

[00083] EEE 17. The apparatus of EEE 15 or 16, wherein the input to the back-bone is processed by a sigmoid before being multiplied by the scaler.

[00084] EEE 18. The apparatus of EEE 15 or 16, wherein the input to the back-bone is
10 processed by a 1x1 separable convolution block before being multiplied by the scaler.

[00085] EEE 19. The apparatus of any of EEES 15-18, wherein the input to the back-bone is processed by a 1x1 separable convolution block followed by a non-linear activation layer before being multiplied by the scaler.

[00086] EEE 20. The apparatus of any one of the EEES 10-19, wherein any 1x3 or 3x1
15 separable CONV is replaced with a corresponding $1 \times M$ or $M \times 1$ separable CONV, where M is an integer larger than 3.

[00087] EEE 21. The apparatus of EEE 20, wherein $M = 5$.

[00088] EEE 22. The apparatus of EEE 20 or 21, further comprising reducing the number of the one or more back-bone blocks.

20 [00089] EEE 23. A non-transitory computer-readable storage medium having stored thereon computer-executable instructions for executing with one or more processors a method in accordance with any one of the EEES 1-7.

[00090] EEE 24. An apparatus comprising a processor and configured to perform any one of the methods recited in EEES 1-7.

25

EQUIVALENTS, EXTENSIONS, ALTERNATIVES AND MISCELLANEOUS

[00091] Example embodiments that relate to loop filtering using neural networks for image and video coding are thus described. In the foregoing specification, embodiments of the present invention have been described with reference to numerous specific details that may
30 vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and what is intended by the applicants to be the invention, is the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation,

element, property, feature, advantage or attribute that is not expressly recited in a claim should limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

1. A method to optimize neural networks with cascaded block networks, the method comprising:
- 5 accessing a first block network comprising:
- a first skip connection between a first start point and a first end point, and two or more first-stage convolutional networks between the first start point and the first end point;
- accessing a second block network subsequent to the first block network comprising:
- 10 a second skip connection between a second start point and a second end point, wherein the second start point coincides with the first end point, and
- two or more second-stage convolutional networks between the second start point and the second end point;
- moving the first start point of the first skip connection to a new position in between the
- 15 two or more first-stage convolutional networks but before the first end point;
- moving the first end point of the first skip connection and the second start point of the second skip connection to the same new position in between the two or more second-stage convolutional networks;
- fusing identical networks between the moved first start point and the moved first end
- 20 point to generate at least one fused convolutional network; and
- generating an output block network comprising:
- the moved first start point, the moved end point,
- a skip connection between the moved start point and the moved end point; and
- at least the one fused convolutional network between the moved first start point
- 25 and the moved first end point.
2. The method of claim 1 wherein the two or more first-stage convolutional networks between the first start point and the first end point comprise: a first 1x1 convolution network (CN), followed by a first non-linear activation, followed by a second 1x1 CN, followed by a first 1x3
- 30 CN, followed by a first 3x1 CN, followed by a third 1x1 CN, and the two or more second-stage convolutional networks between the second start point and the second end point comprise: a fourth 1x1 convolution network (CN), followed by a second non-linear activation, followed by a fifth 1x1 CN, followed by a second 1x3 CN, followed by a second 3x1 CN, followed by a sixth 1x1 CN.

3. The method of claim 2, wherein the start point of the first skip connection is moved after the first 1x1 CN and before the first non-linear activation, the end point of the first skip connection is moved after the fourth 1x1 CN and before the second non-linear activation, and the third
5 1x1 CN is fused with the fourth 1x1 CN to generate a fused 1x1 CN.
4. The method of claim 3, wherein the output block network comprises: the first non-linear activation, followed by the second 1x1 CN, followed by the first 1x3 CN, followed by the first
10 3x1 CN, and followed by the fused 1x1 CN.
5. A method to generate quantization factors to translate a floating-point neural-networks model to an integer-point neural-networks model, the method comprising:
- generating a floating-point performance metric using a floating-point model of a neural network (NN) and a set of test sequences;
- 15 determining initial Q factors for weights and biases to generate a fixed-point model of the NN;
- contacting inference of the fixed-point model using the set of test sequences and computing a fixed-point performance metric of the fixed-point model;
- if the difference between the floating-point performance metric and the fixed-point
20 performance metric is smaller than a performance threshold, then determining an output fixed-point model using Q factors generated up to this step,
- else:
- determining overflow values across all layers of the fixed-point model;
- generating updated Q factors, wherein generating updated Q factors comprises:
- 25 for each overflowed layer, if its overflow is larger than an overflow threshold, then decreasing the Q factors in that layer by a fixed value, else leaving the Q factors in that layer unchanged; and
- returning to the inference step using the updated Q factors.
- 30 6. The method of claim 5, wherein for a 16-bit fixed-point model the initial Q factors are set at Q11.
7. The method of claim 6, wherein Q factors in an overflowed layer are decreased by 1.

8. An apparatus for video loop filtering using neural-network components, the apparatus comprising:

a headblock comprising a series of inputs, each input followed by a 1x1 or 3x3 convolutional network;

5 a fusion and transition (FT) block following the headblock;

a series of one or more back-bone blocks following the fusion and transition block; and

a tail block following the one or more back-bone blocks, wherein the fusion and transition block comprises a 1x1 convolution block with down-sampling followed by a first activation layer (PReLU).

10

9. The apparatus of claim 8, wherein in the FT block, the 1x1 convolution block with down-sampling block is replaced with a 1 x 1 convolution block, and in the headblock each convolutional network is replaced by a 3x3 convolutional network with down-sampling.

15 10. An apparatus for video loop filtering using neural-network components, the apparatus comprising:

a headblock comprising a series of inputs, each input followed by a convolutional network;

a fusion and transition (FT) block following the headblock;

20 a series of one or more back-bone blocks following the fusion and transition block; and

a tail block following the one or more back-bone blocks, wherein the FT block comprises:

a first 1x1 convolutional (CONV) block, followed by

a first activation layer (PReLU), followed by

25 a 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling, followed by

a second activation layer (PReLU), wherein the CPD block comprises:

a second 1x1 CONV block, followed by

a 1x3 separable CONV with down sampling, followed by

a 3x1 separable CONV with down sampling, followed by

30 a third 1x1 CONV block.

11. The apparatus of claim 10, wherein the FT block is replaced by a second FT block, the second FT block comprising:

the first activation layer (PReLU), followed by

the 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling, followed by
the second activation layer.

5 12. The apparatus of claim 10, wherein the FT block is replaced by a third FT block, the third FT block comprising:

the first activation layer (PReLU), followed by
a second 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling.

10 13. The apparatus of claim 12, wherein the second 3 x 3 canonic polyadic decomposition (CPD) block with down-sampling comprises:

the second 1x1 CONV block, followed by
a second 1x3 separable CONV with down sampling, followed by
a second 3x1 separable CONV with down sampling.

15

14. An apparatus for video loop filtering using neural-network components, the apparatus comprising:

a headblock comprising a series of inputs, each input followed by a convolutional network;

20

a fusion and transition (FT) block following the headblock;

a series of one or more back-bone blocks following the fusion and transition block; and

a tail block following the one or more back-bone blocks, wherein after the last back-bone block the tail block comprises:

an activation layer (PReLU), followed by

25

a 3x3 convolutional layer.

15. The apparatus of claim 14, wherein a back-bone comprises:

an activation layer (PReLU), followed by

a first 1x1 CONV block, followed by

30

a 1x3 separable CONV, followed by

a 3x1 separable CONV, followed by

a second 1x1 CONV block, followed by

a scaler (802), wherein the scaler multiplies an input to the back-bone with the output (804) of the second 1x1 CONV block, followed by

an adder, wherein the adder adds an input to the back-bone with the output of the scaler.

16. The apparatus of claim 15, wherein the positions of the activation layer (PReLU) and the first 1x1 CONV block are swapped.

5

17. The apparatus of claim 15 or 16, wherein the input to the back-bone is processed by a sigmoid before being multiplied by the scaler.

18. The apparatus of claim 15 or 16, wherein the input to the back-bone is processed by a 1x1
10 separable convolution block before being multiplied by the scaler.

19. The apparatus of any one of claims 15-18, wherein the input to the back-bone is processed by a 1x1 separable convolution block followed by a non-linear activation layer before being multiplied by the scaler.

15

20. The apparatus of any one of the claims 10-19, wherein any 1x3 or 3x1 separable CONV is replaced with a corresponding 1 x M or M x 1 separable CONV, where M is an integer larger than 3.

20 21. The apparatus of claim 20, wherein $M = 5$.

22. The apparatus of claim 20, further comprising reducing the number of the one or more back-bone blocks.

25 23. A non-transitory computer-readable storage medium having stored thereon computer-executable instructions for executing with one or more processors a method in accordance with any one of the claims 1-7.

24. An apparatus comprising a processor and configured to perform any one of the methods
30 recited in claims 1-7.

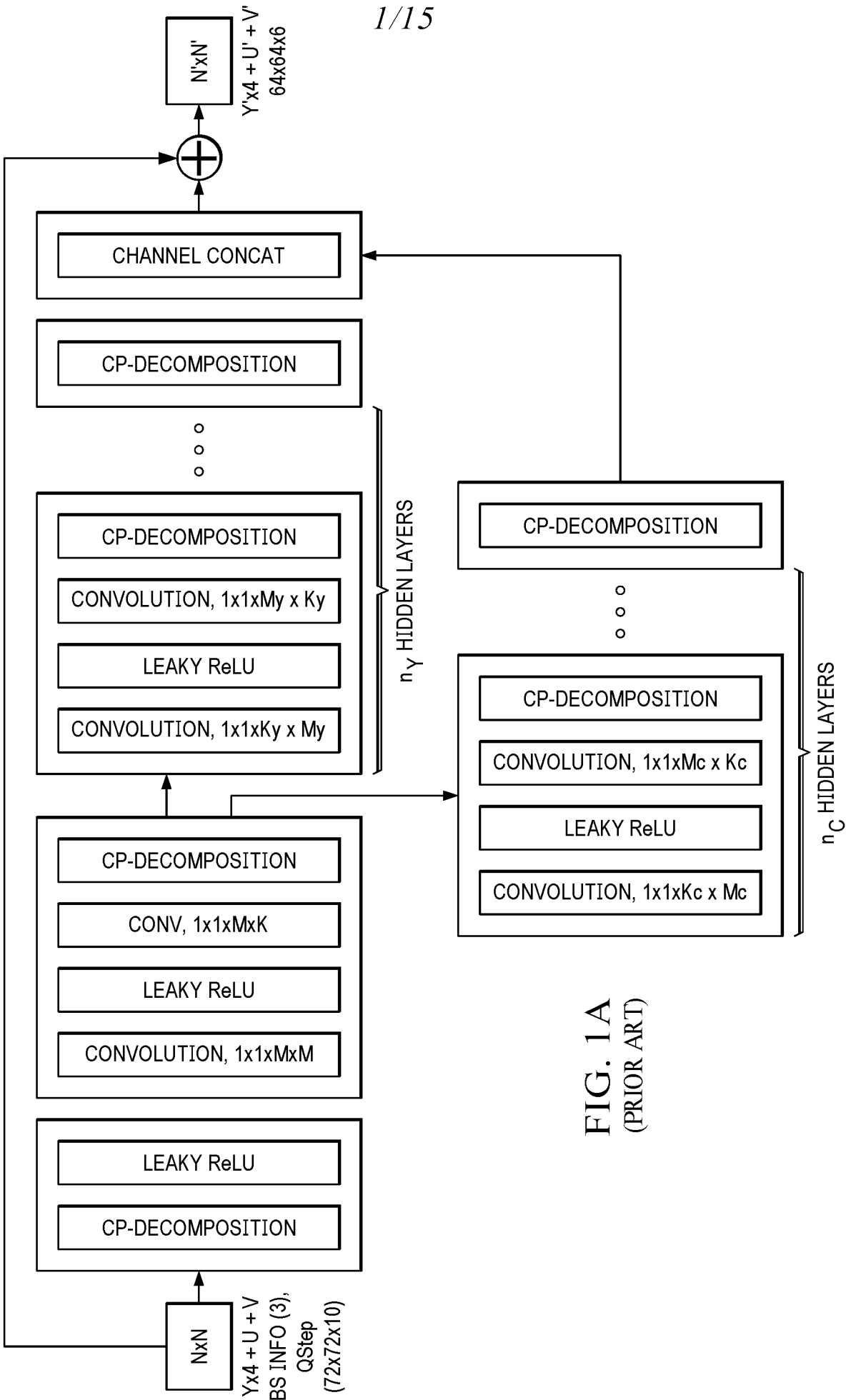
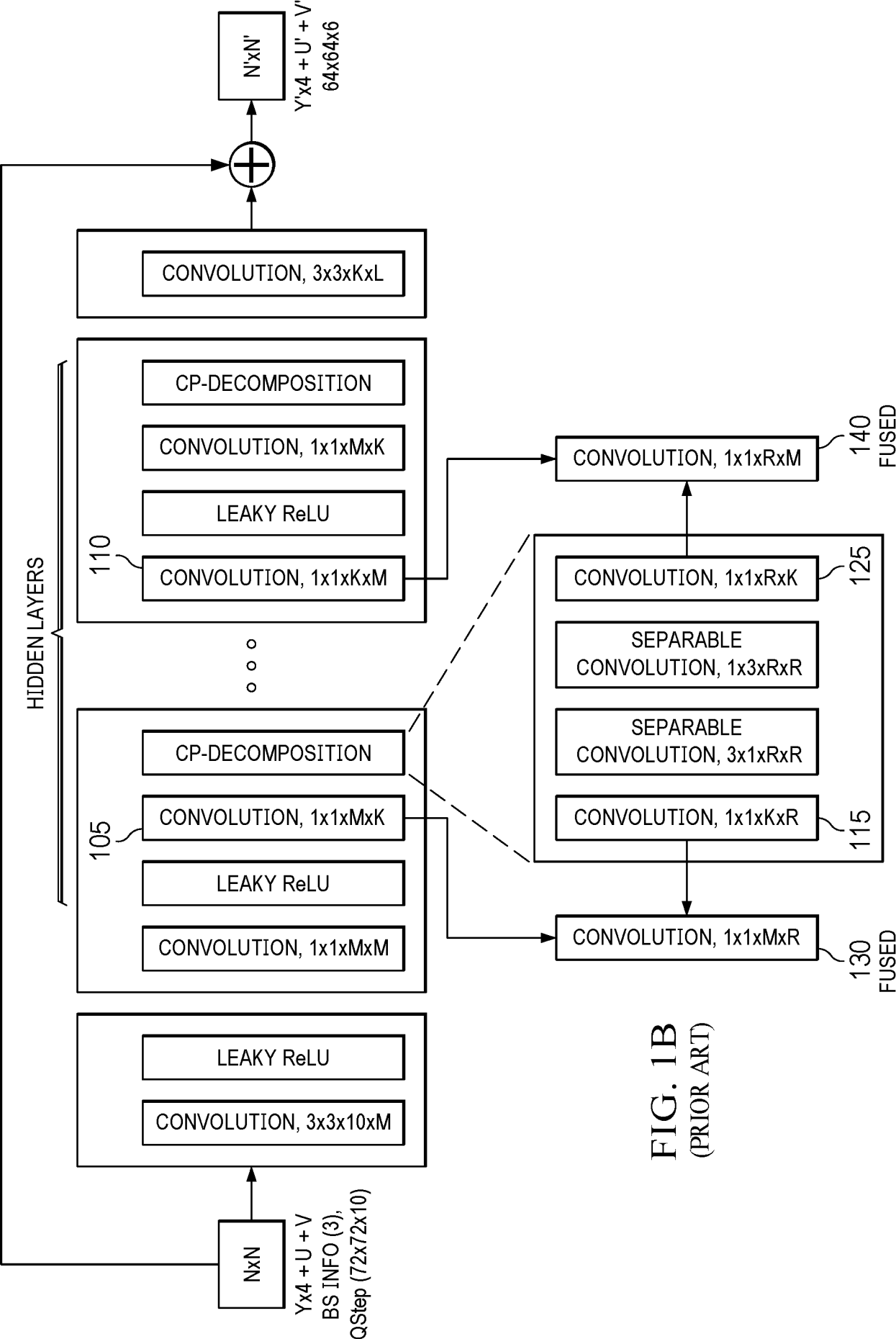
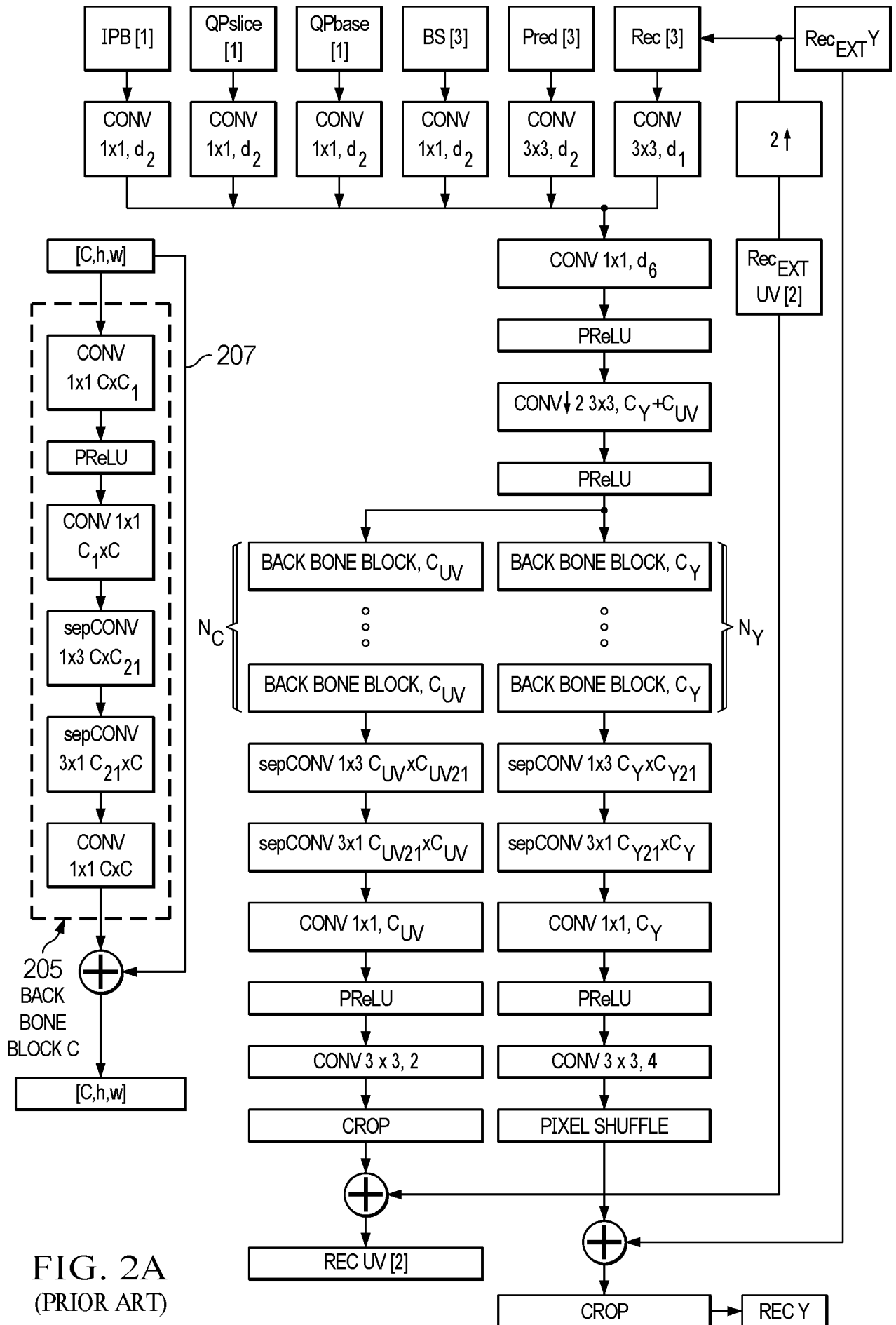


FIG. 1A
(PRIOR ART)



3/15

FIG. 2A
(PRIOR ART)

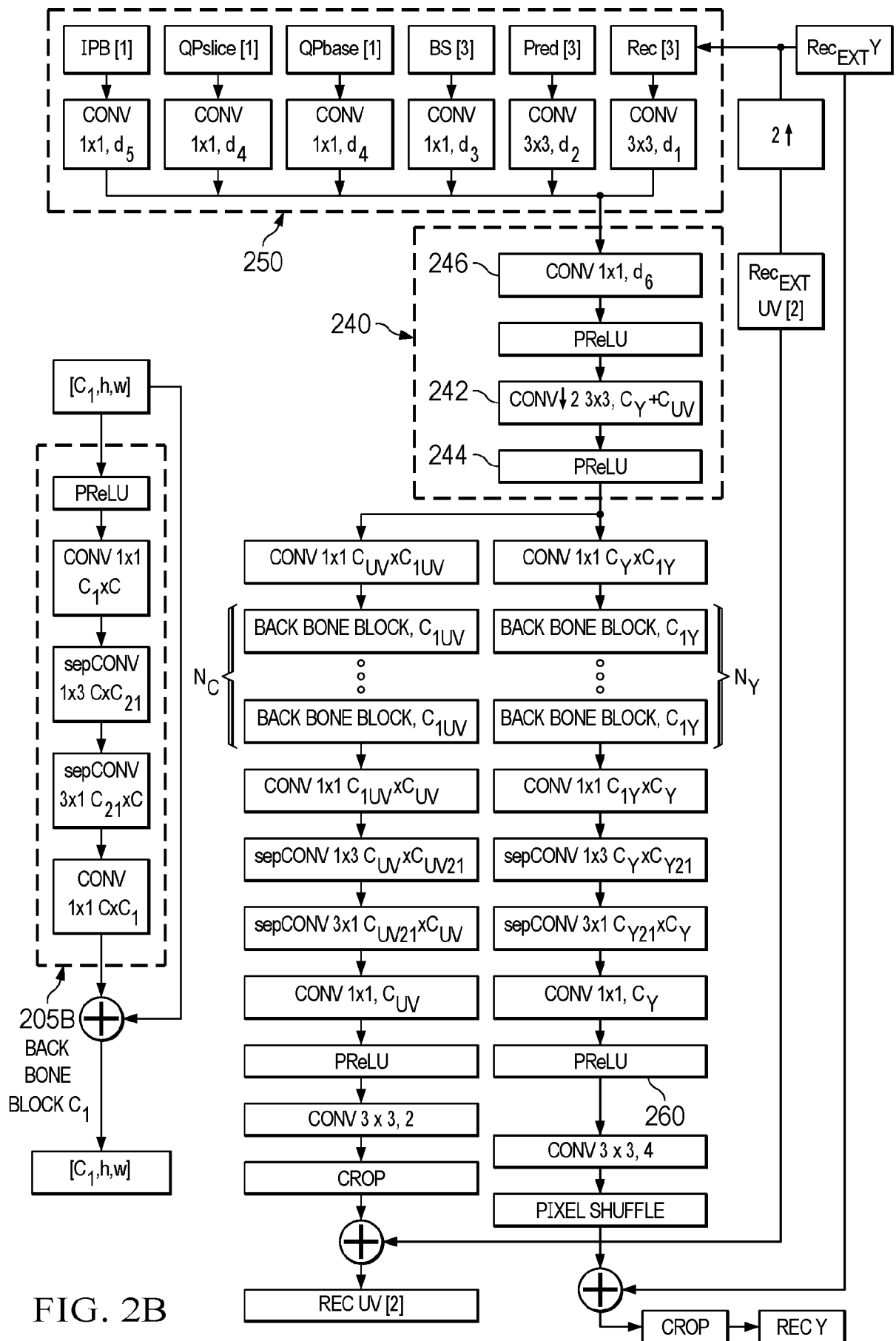


FIG. 2B

5/15

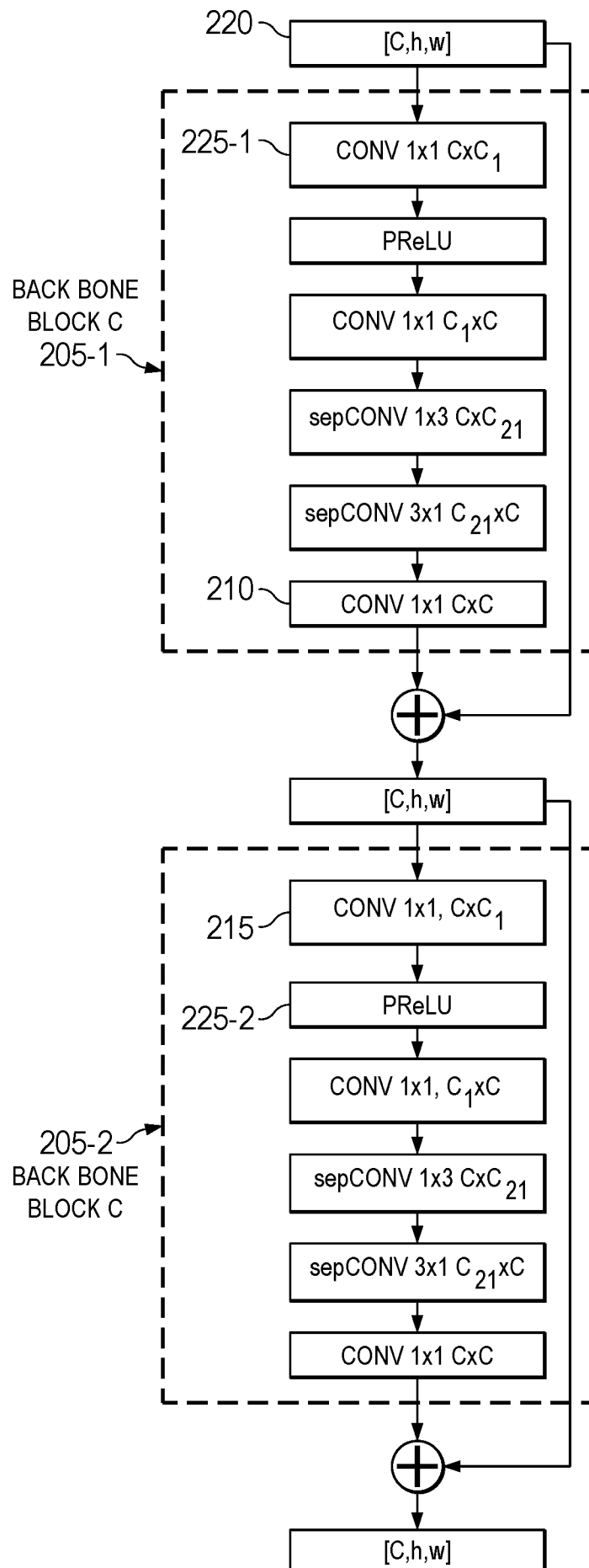


FIG. 3A

6/15

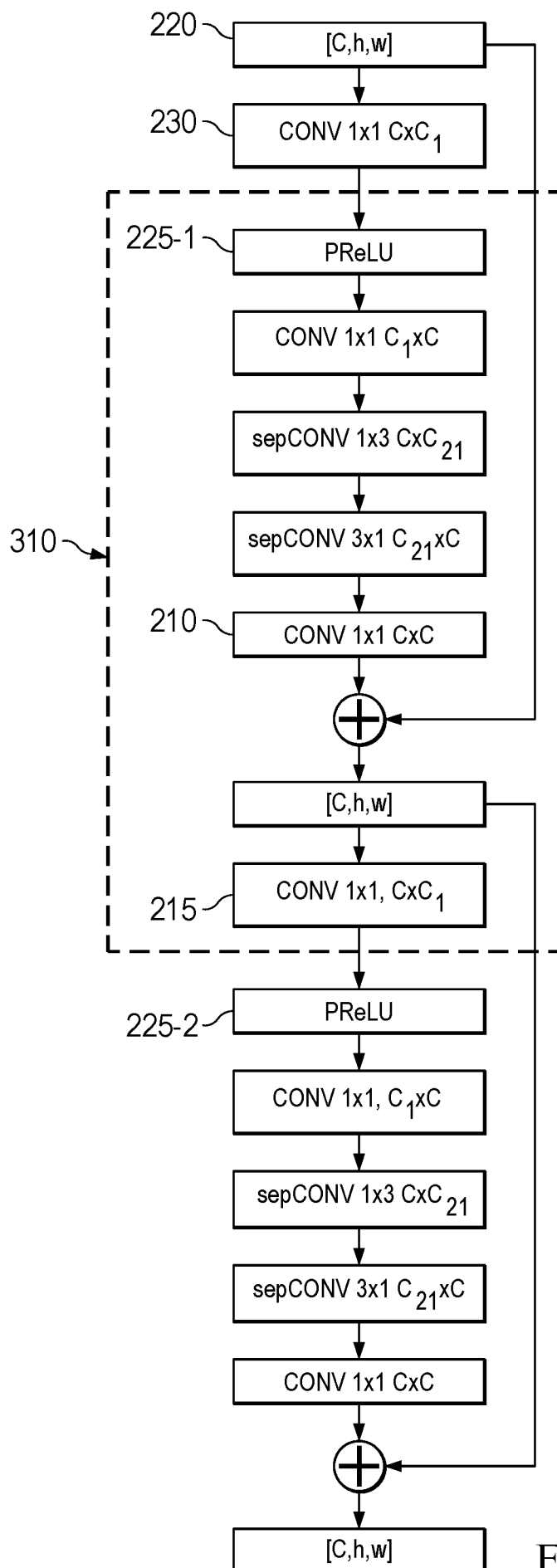


FIG. 3B

7/15

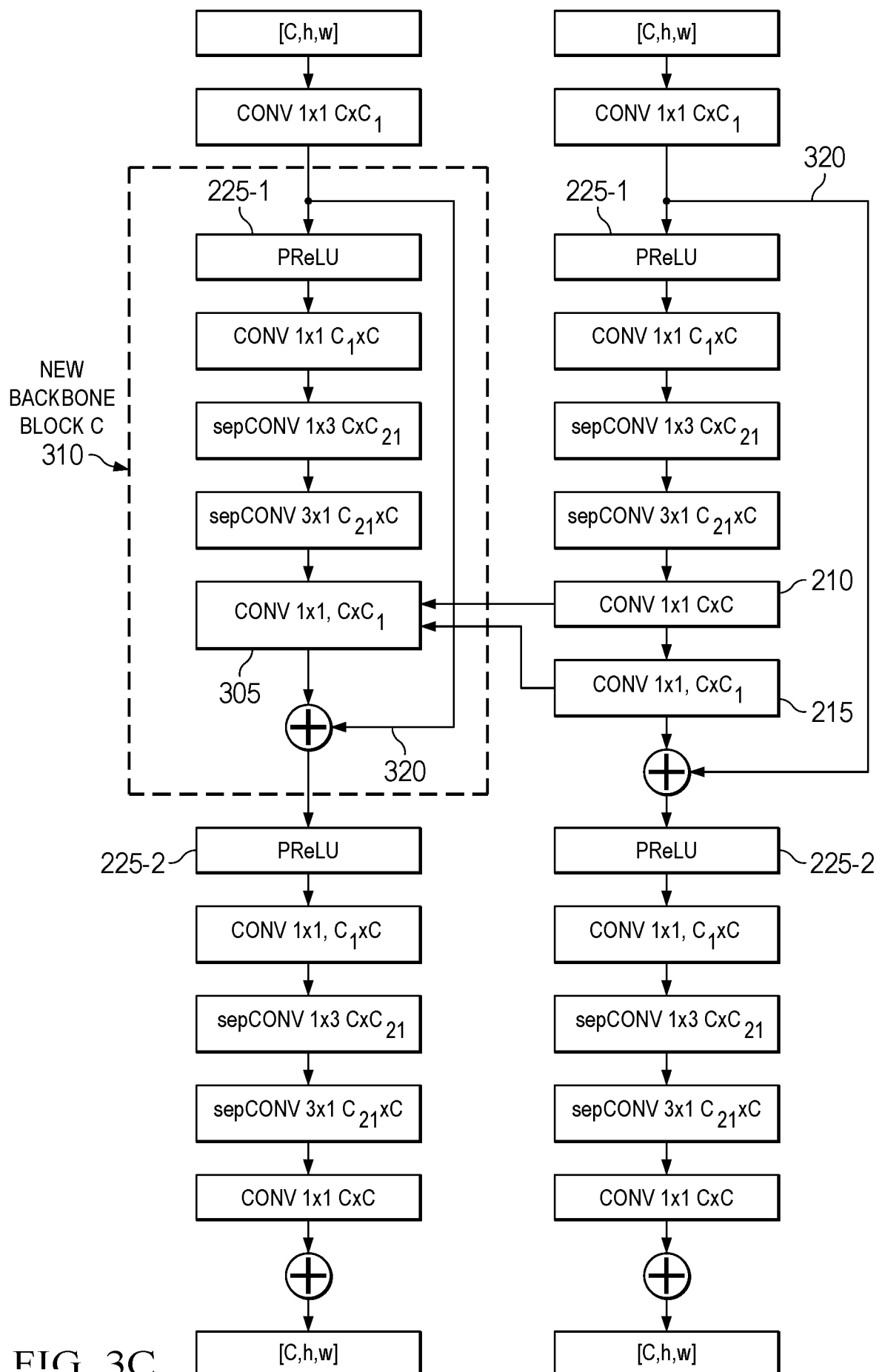


FIG. 3C

8/15

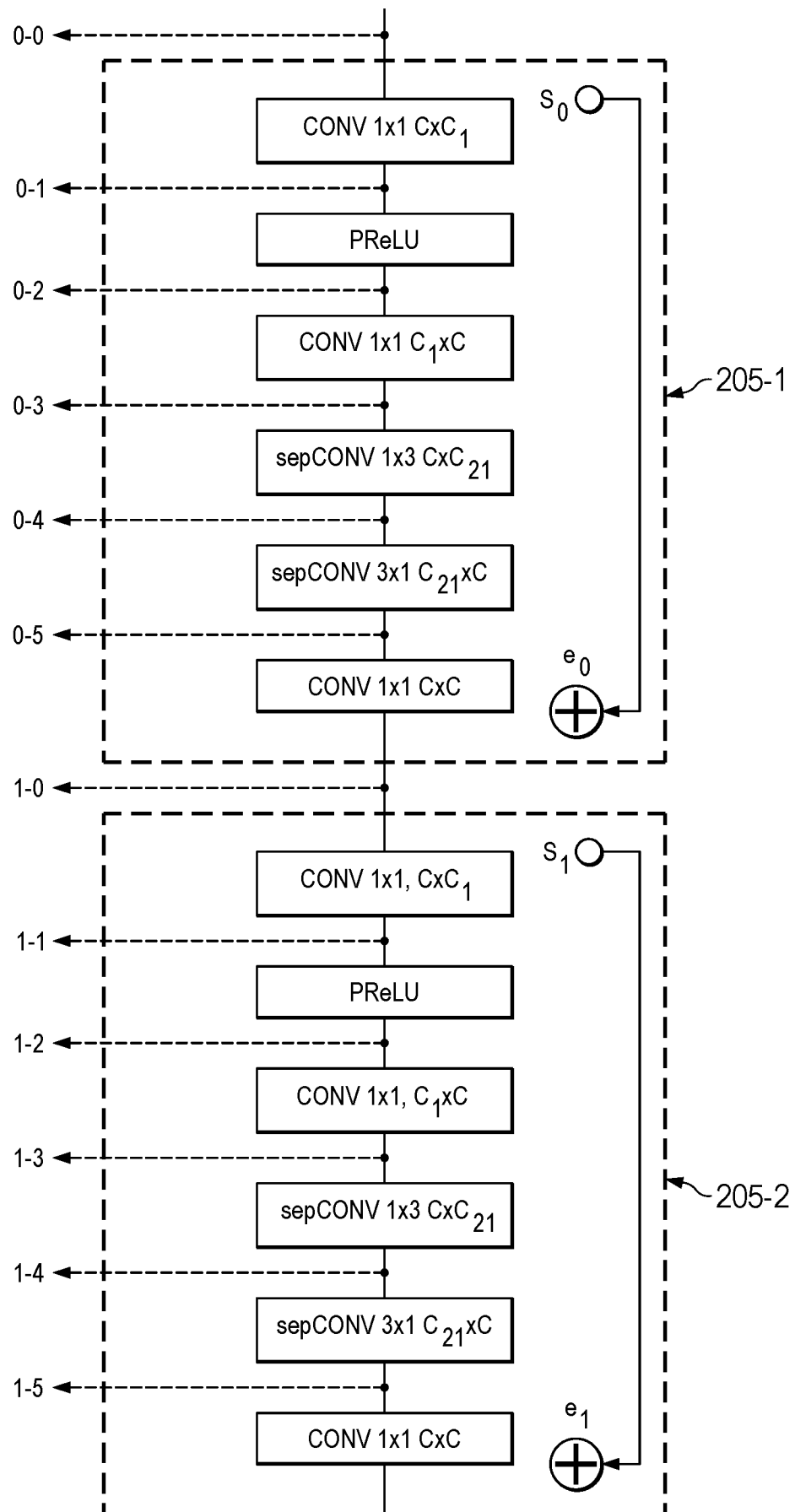


FIG. 4

9/15

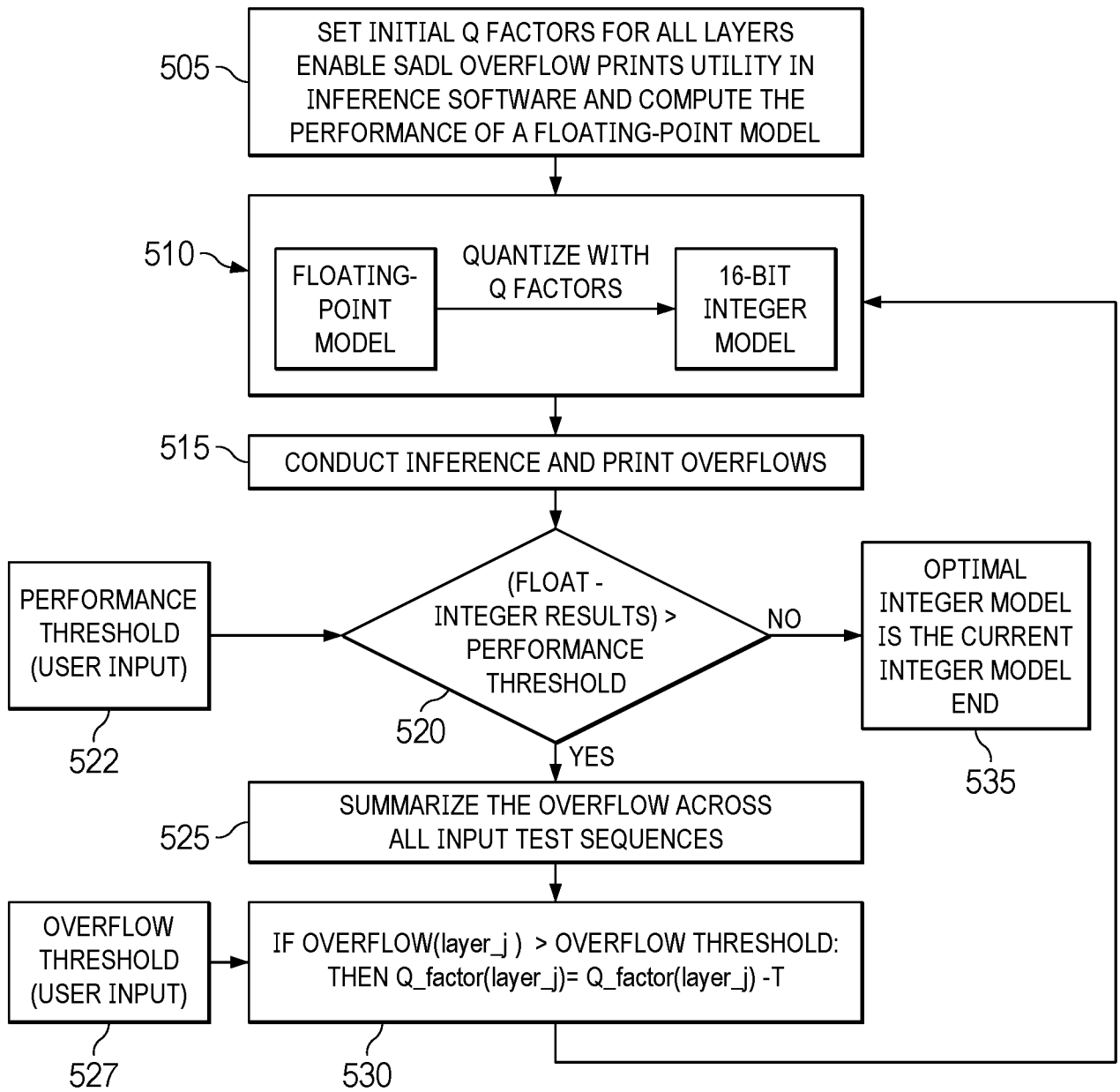


FIG. 5

10/15

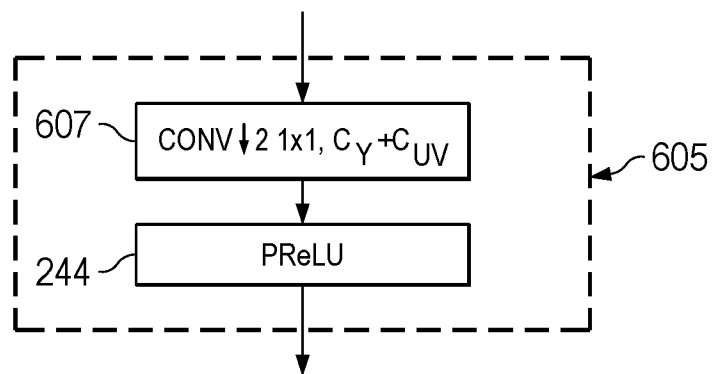


FIG. 6A

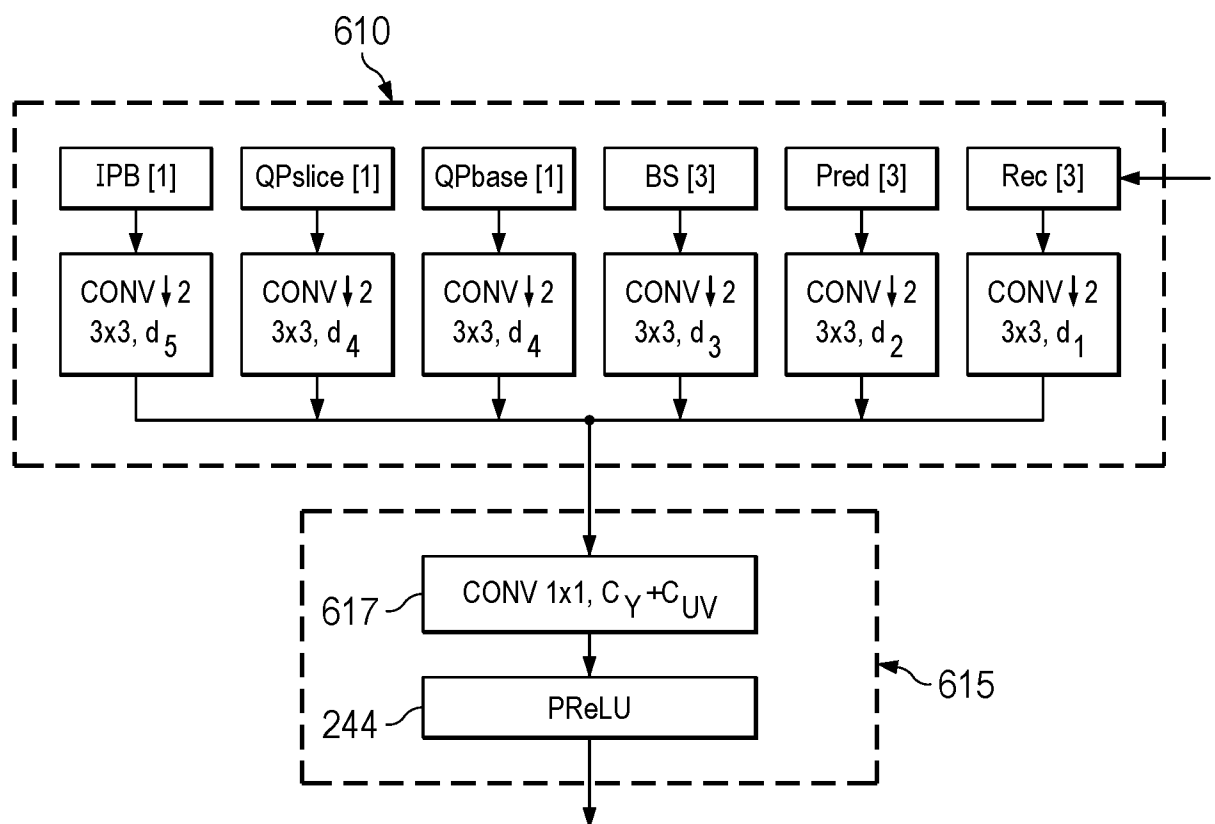


FIG. 6B

11/15

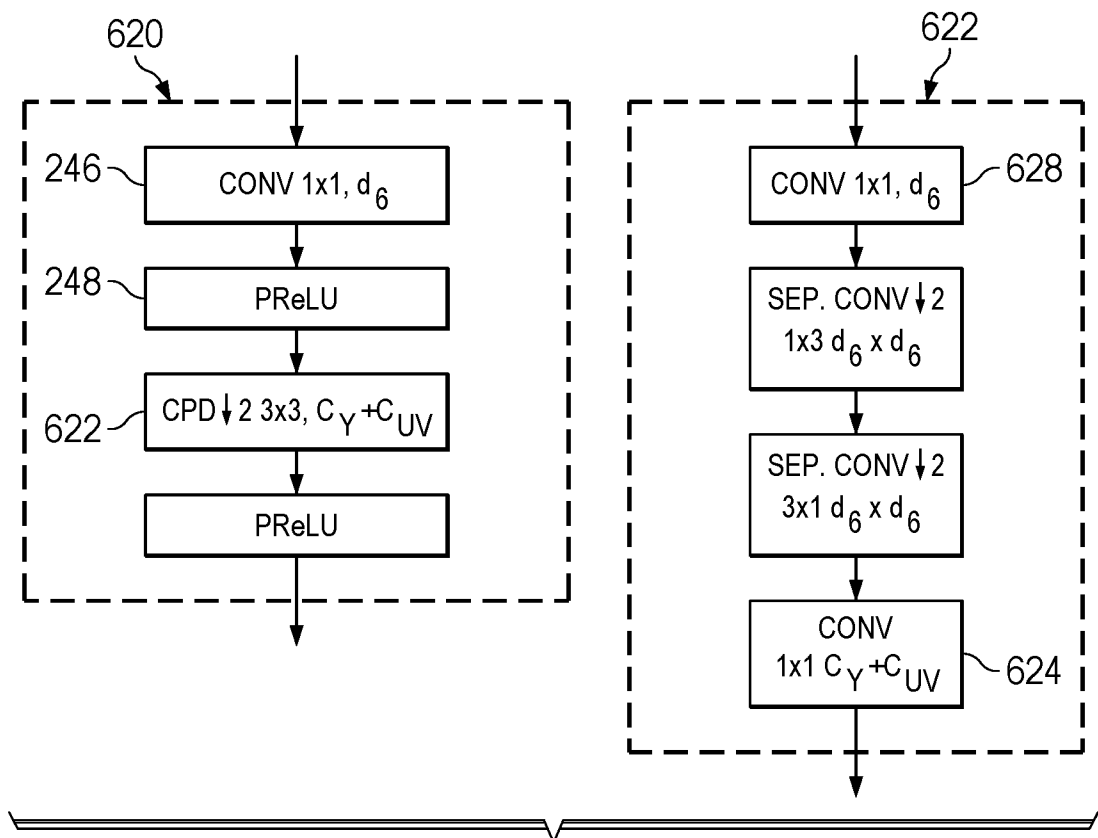


FIG. 6C

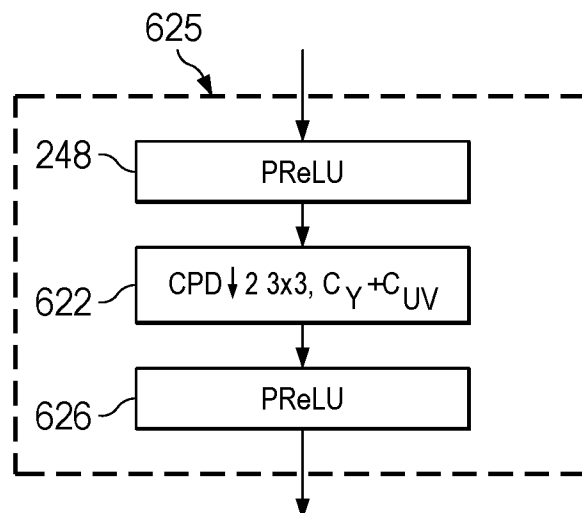


FIG. 6D

12/15

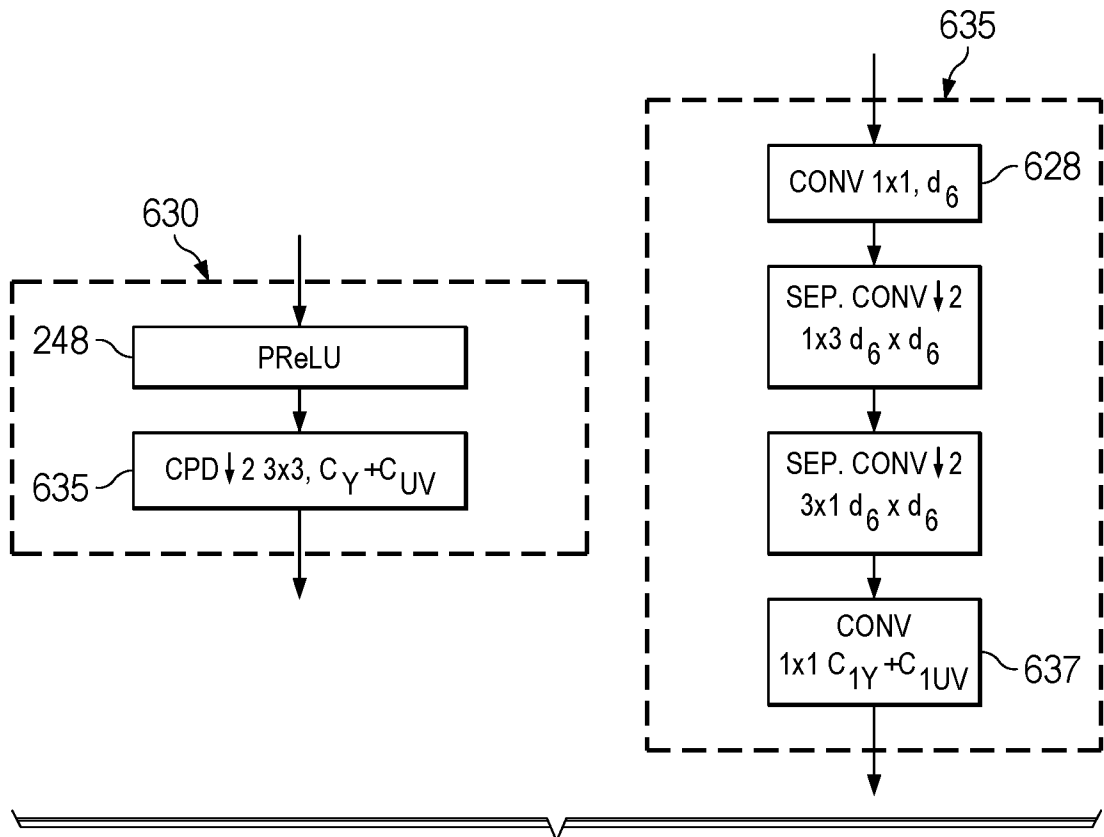


FIG. 6E

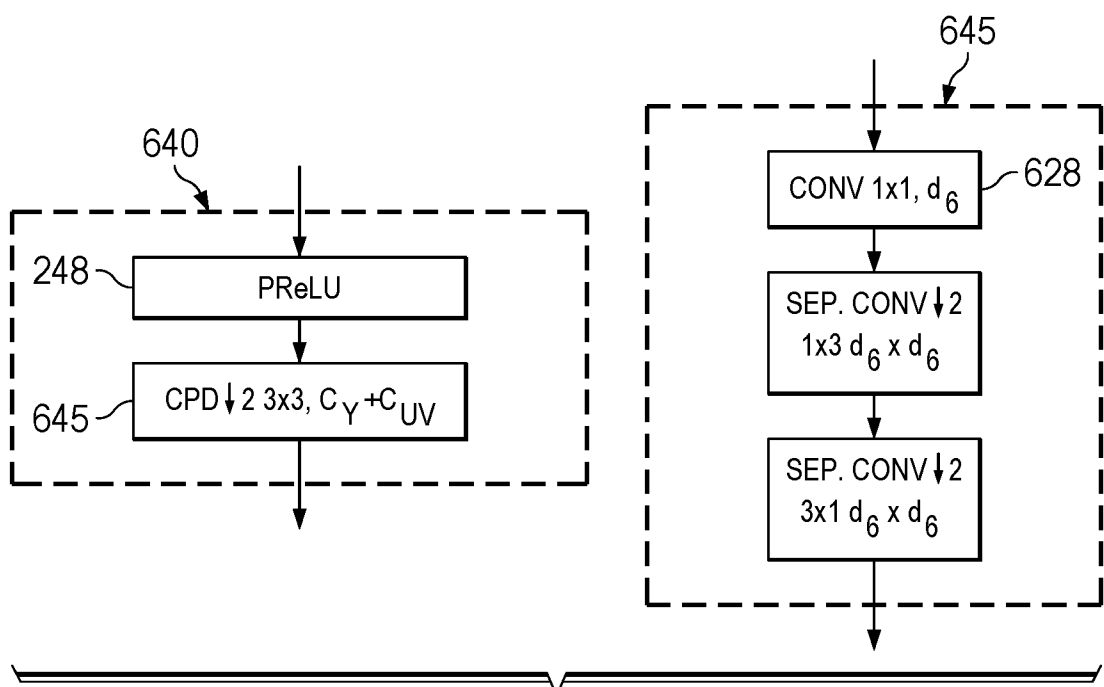


FIG. 6F

13/15

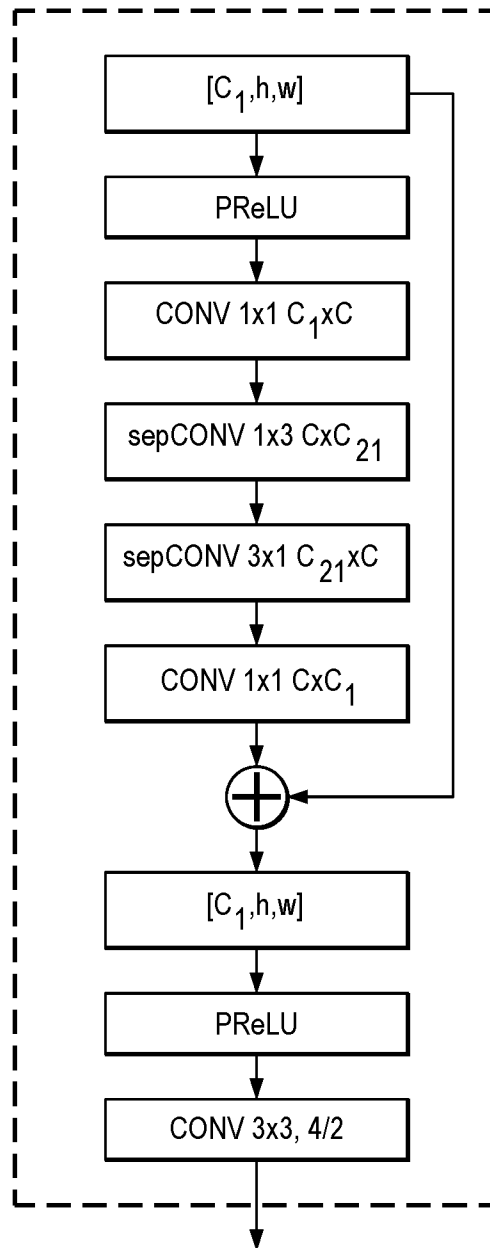


FIG. 6G

14/15

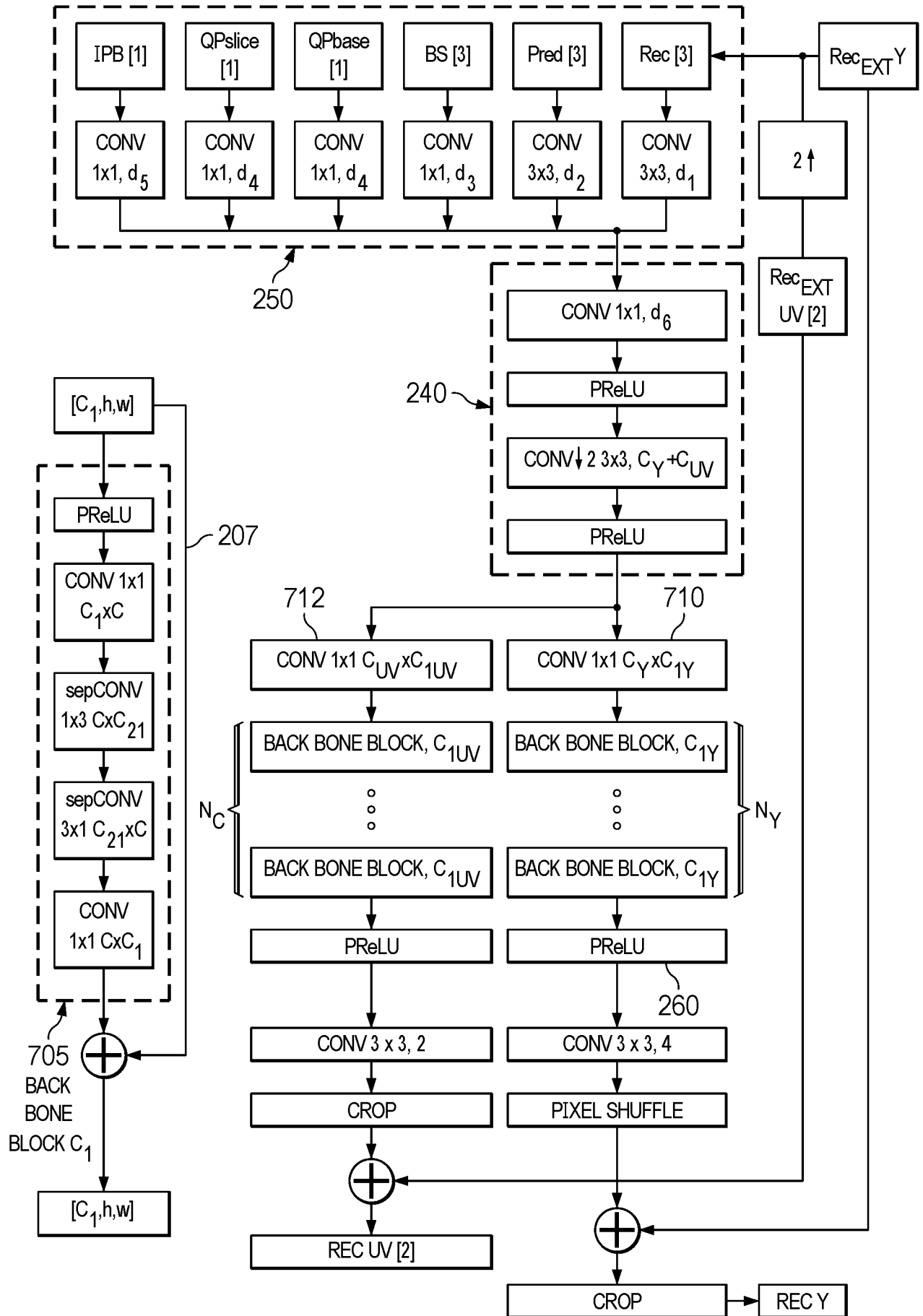


FIG. 7

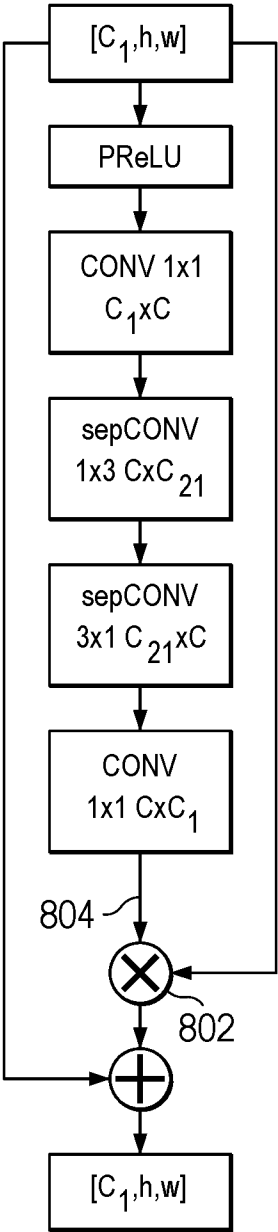


FIG. 8A

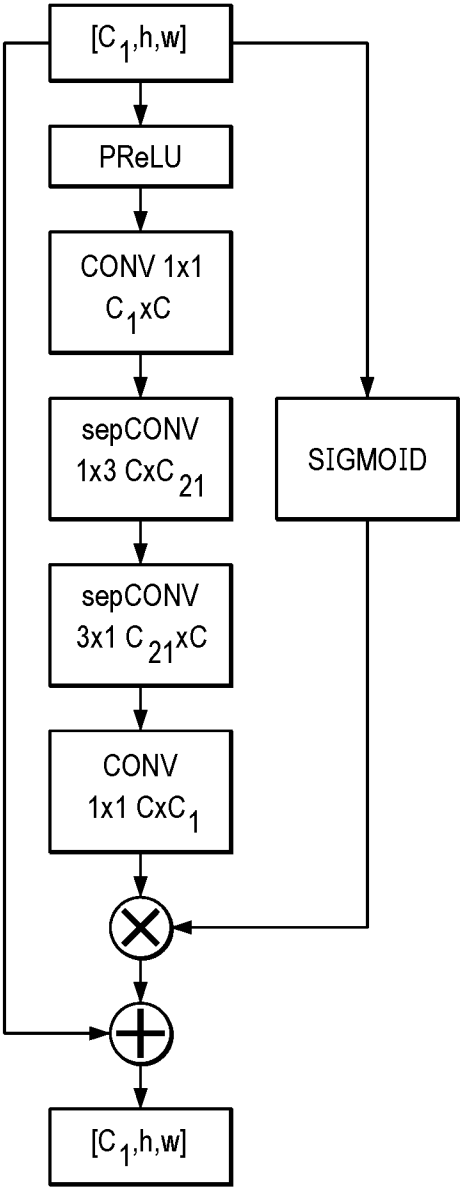


FIG. 8B