**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG2020/m54468**

**July 2020, OnLine**

|  |  |
| --- | --- |
| **Source:** | **InterDigital Communications, Inc.** |
| **Status:** | **Input contribution** |
| **Title:** | **[NNR] Syntax and mechanism for weight tensor decomposition** |
| **Authors:** | **Fabien Racapé, Shahab Hamidi-Rad, Swayambhoo Jain** |

# Introduction

This contribution presents a mechanism and the syntax elements required to support the reconstruction of decomposed tensors of parameters. The proposed syntax has been added on top of the Working Draft 4 of Compression of neural networks for multimedia content description and analysis [1]. Please refer to the attached edited working draft for exact text changes.

# Decoded Tensor Buffer

The current NNR Working Draft 4 [1], decomposes the bitstream into NNR Units as shown in Figure 1. These units contain either Headers and High-Level Syntax parameter sets or payload corresponding to compressed tensor parameters.



Figure 1: NNR bitstream data structure

In the current architecture, a compressed data unit contains one tensor at a time. However, in the case of tensor decomposition, reconstructing the tensors in their original shape might be required, which requires to access two decoded units.

For instance, Low Rank approximations represent an original matrix of weights as a product:

$$W=G H^{T}$$

where ***G*** is a $ m ×r$ matrix and ***H***$ $is $n × r$ matrix that can be derived from an SVD decomposition, *r* corresponds to the chosen rank for factorizing the input ***W*** of size $m×n$.

Since ***G*** and ***H*** need to be multiplied at the decoder to reconstruct the tensor in its original shape, but belong to different units, this contribution proposes to introduce a Decoded Tensor Buffer (DTB) that enables such reconstruction at the decoder as well as the mechanism to output or store decoded tensors.

The following subsections 6.3.2 and 6.3.3 are proposed to be added to the specification text (the numbers have been kept in line with the current working draft).

In the following, the added text is highlighted in yellow.

**6.3.1 Tensor output**

This process is invoked once per NNR compressed payload, after decoding of the unit header nnr\_compressed\_data\_unit\_header and the compressed payload.

The output of the current tensor is specified as follows:

* If output\_original\_graph is equal to 0 or if lps\_tensor\_decomposition\_flag is equal to 0, or if nnr\_decomposition\_tensor\_type is equal to “TENSOR\_OTHER”, the current tensor is output
* Otherwise if there are no tensors with a ref\_id specifying the same layer in the DTB, add the current tensor into the DTB. No tensor is output.
* Otherwise (the current nnr\_decomposition\_tensor\_type specifies a tensor of type “TENSOR\_G” or “TENSOR\_H” and there exists a tensor with ref\_id specifying the same layer in the DTB), invoke the reconstruction of the tensor in its original shape, as specified in clause 6.3.2, passing both the current tensor of type “TENSOR\_G” or “TENSOR\_H”, respectively, and its corresponding tensor in the DTB, of type “TENSOR\_H” or “TENSOR\_G”, respectively. The latter is deleted from the DTB. The returned tensor is output.

**6.3.2 Reconstruction of tensors in their original shape**

This process is invoked after the decoding a tensor of type “TENSOR\_G” or “TENSOR\_H” and the corresponding tensor with ref\_id specifying the same layer is present in the DTB, as specified in clause 6.3.1.

The inputs to this process are

* A tensor tensor\_h[ ] of type “TENSOR\_H”
* The tensor\_dimensions\_h[ ] corresponding to the dimensions of tensor\_h[ ] , as defined by its decoded syntax tensor\_dimensions from the corresponding nnr\_compressed\_data\_unit\_header
* A tensor tensor\_g[ ] of type “TENSOR\_G”.
* An array tensor\_dimensions\_g[ ] corresponding to the dimensions of tensor\_g[ ], as defined by its decoded syntax tensor\_dimensions
* The values tensor\_reconstruction\_mode and tensor\_reconstruction\_additional\_info from the layer parameter set of the tensor just decoded header.

Output of this process is the current tensor array\_w in its original shape, prior to applying tensor decomposition.

Table 5: Specification of the reconstruction mode

|  |  |  |
| --- | --- | --- |
| **tensor\_reconstruction\_mode**  | **Name of tensor\_reconstruction\_mode**  | **Description** |
| 0 | NNR\_FC | Fully connected layer |
| 1 | NNR\_CONV | Convolutional layer |
| 2 | NNR\_DWCONV | Depth-wise convolutional layer |
| 3-7 | NNR\_URM | unspecified reconstruction modes |

if ( tensor\_reconstruction\_mode == NNR\_FC )

array\_w = MatrixProd( tensor\_g, tensor\_h )

else if ( tensor\_reconstruction\_mode== NNR\_CONV ) {

rank = tensor\_dimensions\_g [3]

wShape = [tensor\_dimensions\_g [0], tensor\_dimensions\_g [1], tensor\_dimensions\_g [2], tensor\_dimensions\_h [3]]

prod = MatrixProd( TensorReshape (g, [-1, rank]), TensorReshape (h, [rank, -1])

array\_w = TensorReshape (prod, wShape)

}

else if ( tensor\_reconstruction\_mode== NNR\_DWCONV ) {

kernel = tensor\_reconstruction\_additional\_info[0]

wShape = [kernel, kernel, -1,1]

array\_w = TensorReshape (MatrixProd( g, h ), wShape)

}

The following operation definitions are required:

MatrixProd( array\_name\_1[], array\_name\_2[] ) returns the matrix product of array\_name\_1 by array\_name\_2.

TensorReshape( array\_name[], tensor\_dimension[]) returns the reshaped tensor array\_name[] with the specified tensor\_dimension[], without changing its data.

# High Level Syntax

## HLS for Tensor reconstruction

In terms of HLS, the following sets of parameters need to be added/updated:

**8.2.4.6 NNR compressed data unit header syntax**

|  |  |
| --- | --- |
| nnr\_compressed\_data\_unit\_header( ) { | **Descriptor** |
|  **nnr\_layer\_parameter\_set\_id** | u(8) |
|  **nnr\_compressed\_data\_unit\_payload\_type** | u(5) |
|  **nnr\_multiple\_topology\_elements\_present\_flag** | u(1) |
|  **nnr\_decompressed\_data\_format\_present\_flag** | u(1) |
|  **input\_parameters\_present\_flag** | u(1) |
|  if (nnr\_multiple\_topology\_elements\_present\_flag = = 1) |  |
|  topology\_elements\_ids\_list() |  |
|  else |  |
|  **ref\_id** | st(v) |
|  if (nnr\_compressed\_data\_unit\_payload\_type = = NNR\_PT\_CB\_FLOAT32) { |  |
|  **codebook\_zero\_offset** | u(8) |
|  **codebook\_size** | u(16) |
|  For ( j = 0 ; j < codebook\_size; j++ ) { |  |
|  **codebook**[j] | flt(32) |
|  } |  |
|  } |  |
|  if (lps\_tensor\_decomposition\_flag == 1) { |  |
|  **nnr\_decomposition\_tensor\_type** | u(2) |
|  } |  |
|  if (nnr\_decompressed\_data\_format\_present\_flag = = 1) |  |
|  **nnr\_decompressed\_data\_format** | u(7) |
|  if (input\_parameters\_present\_flag = = 1) { |  |
|  **tensor\_dimensions\_flag** | u(1) |
|  **cabac\_unary\_length\_flag** | u(1) |
|  if (tensor\_dimensions\_flag = = 1) |  |
|  tensor\_dimensions() |  |
|  If (cabac\_unary\_length\_flag = = 1) |  |
|  **cabac\_unary\_length** | u(8) |
|  } |  |
|  byte\_alignment() |  |
| } |  |

Syntax element definitions:

**nnr\_layer\_parameter\_set\_id** specifies the value of lps\_layer\_parameter\_set\_id for the compressed unit in use. The value of unit\_layer\_parameter\_set\_id shall be in the range of 0 to 63, inclusive.

**nnr\_decomposition\_tensor\_type** specifies the tensor type in the case of tensor decomposition, as defined in clause 6.2

**8.2.5.2 NNR model parameter set payload syntax**

|  |  |
| --- | --- |
| nnr\_model\_parameter\_set\_payload( ) { | **Descriptor** |
|  **mps\_model\_parameter\_set\_id** | u(4) |
|  **topology\_carriage\_flag** | u(1) |
|  **sparsification\_flag**  | u(1) |
|  **mps\_tensor\_decomposition\_flag** | u(1) |
|  **quantization\_method\_flags** | u(6) |
|  if ((quantization\_method\_flags & NNR\_QSU) = = NNR\_QSU) { |  |
|  **qp\_density** | u(3) |
|  **quantization\_parameter** | i(13) |
|  } |  |
|  If (sparsification\_flag = = 1) { |  |
|  sparsification\_performance\_map() |  |
|  }  |  |
|  if (mps\_tensor\_decomposition\_flag) { |  |
|  **output\_original\_graph** | u(1) |
|  if (output\_original\_graph)  |  |
|  decomposition\_performance\_map() |  |
|  **mps\_max\_dec\_tensor\_buffering\_minus1** | u(6) |
|  } |  |
|  **ctu\_partition\_flag** | u(1) |
|  if(ctu\_partition\_flag){ |  |
|  **max\_ctu\_dim\_flag** | u(2) |
|  **nnr\_reserved\_zero\_5bits** | u(5) |
|  }else{ |  |
|  **nnr\_reserved\_zero\_7bits** | u(7) |
|  } |  |
| } |  |

**mps\_model\_parameter\_set\_id** provides an identifier for the MPS for reference by other syntax elements. The value of mps\_model\_parameter\_set\_id shall be in the range of 0 to 15, inclusive.

**decomposition\_flag** equal to 1specifies that tensor decomposition was applied to at least one layer of the model.

**output\_original\_graph** equal to 1 specifies that the decoder outputs the tensors of weights in their original shape when tensor decomposition is used.

**mps\_max\_dec\_tensor\_buffering\_minus1** plus 1 specifies the maximum required size of the decoded tensor buffer for the NNR model, in units of tensor storage buffers. The value of mps\_max\_dec\_tensor\_buffering\_minus1 shall be in the range of 0 to 63

**8.2.5.3 NNR layer parameter set unit payload syntax**

|  |  |
| --- | --- |
| nnr\_layer\_parameter\_set\_unit\_payload() { | **Descriptor** |
|  **lps\_model\_parameter\_set\_id** | u(4) |
|  **lps\_layer\_parameter\_set\_id** | u(6) |
|  **independently\_decodable\_flag** | u(1) |
|  **lps\_tensor\_decomposition\_flag** | u(1) |
|  **sparsification\_flag**  | u(1) |
|  **quantization\_method\_flags** | u(6) |
|  If ((quantization\_method\_flags & NNR\_QSU) = = NNR\_QSU) { |  |
|  **quantization\_step\_size** | u(8) |
|  } |  |
|  If ((quantization\_method\_flags & NNR\_QCB) = = NNR\_QCB) { |  |
|  quantization\_map() |  |
|  } |  |
|  if (lps\_tensor\_decomposition\_flag) { |  |
|  **tensor\_reconstruction\_mode** | u(3) |
|  **tensor\_reconstruction\_additional\_info\_count** | u(5) |
|  for (j = 0; j < (tensor\_reconstruction \_additional\_info\_count); j++ ) { |  |
|  **tensor\_reconstruction\_additional\_info[j]** | u(16) |
|  } |  |
|  If (sparsification\_flag = = 1) { |  |
|  sparsification\_performance\_map() |  |
|  }  |  |
| } |   |

**lps\_model\_parameter\_set\_id** specifies the value of the mps\_model\_parameter\_set\_id of the active LPS. . The value of lps\_model\_parameter\_set\_id shall be in the range of 0 to 15, inclusive.

**lps\_layer\_parameter\_set\_id** provides an identifier for the LPS for reference by other syntax elements. The value of lps\_layer\_parameter\_set\_id shall be in the range of 0 to 63, inclusive.

**lps\_tensor\_decomposition\_flag** equal to 1 specifies that tensor decomposition is used for this layer.

**tensor\_reconstruction\_mode** specifies the mode which is used to reconstruct the current tensor in its original shape from decomposed decoded tensors as defined in subclause 9.1.1.

**tensor\_reconstruction\_additional\_info\_count** specifies the number of required parameters to perform the reconstruction of decomposed tensors

**tensor\_reconstruction\_additional\_info[ i ]** specifies an array of parameters which are required for reconstructing decomposed tensors. (For example, in the case of a depth-wise convolutional layer, tensor\_reconstruction\_additional\_info\_count can be set to 1 and tensor\_reconstruction\_additional\_info[ 0 ] specifies the kernel size of the convolution).

## Decomposition performance map

As for sparsification methods, the receiver can benefit from additional information on how the decomposition impacts the performance of the network.

To this end, it is proposed to include the following Decomposition Performance Map, which is accessed via the Model Parameter set, as shown in the introduced section 8.2.5.2.

decomposition\_performance\_map() is defined as follows:

|  |  |
| --- | --- |
| decomposition\_performance\_map () { | **Descriptor** |
|  **count\_thresholds** | u(8) |
|  for (i = 0; i < (count\_thresholds-1); i++ ) { |  |
|  **mse\_threshold[i]** | flt(32) |
|  **nn\_accuracy[i]** | flt(32) |
|  **nn\_reduction\_ratio[i]** | flt(32) |
|  **count\_classes[i]** | u(16) |
|  for (j = 0; j < (count\_classes-1); j++ ) { |  |
|  **nn\_class\_accuracy[i][j]** | flt(32) |
|  } |  |
|  } |  |
| } |   |

**decomposition\_performance\_map()** specifies a mapping between different Mean Square Error (MSE) thresholds between the decomposed tensors and their original version and resulting NN inference accuracies. The resulting accuracies are provided separately for different aspects or characteristics of the output of the NN. For a classifier NN, each MSE threshold is mapped to separate accuracies for each class, in addition to an overall accuracy which considers all classes. Classes are ordered based on the neural network output order, i.e, the order specified during training.

**count\_thresholds** specifies the number of decomposition MSE thresholds.

**decomposition\_threshold** specifies an array of MSE thresholds which are applied to derive the ranks of the different tensors of weights.

**nn\_accuracy** specifies the overall accuracy of the NN (e.g., classification accuracy by considering all classes).

**nn\_reduction\_ratio[i]** specifies the ratio between the total number of parameters after tensor decomposition of the whole model and the number of parameters in the original model

**count\_classes** specifies number of classes for which separate accuracies are provided for each decomposition thresholds.

**nn\_class\_accuracy** specifies an array of accuracies for a certain class, when a certain decompsition threshold is applied.

# References

[1] “N19225 - Working Draft 4 of Compression of neural networks for multimedia content description and analysis.” International Organization for Standardization ISO/IEC JTC1/SC29/WG11, Apr. 2020.

# Patent rights declaration(s)

**InterDigital Communications, Inc. may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**