**Source:** **Interdigital Finland Oy**

**Title: [FS\_AI4Media] pCR on intermediate data compression editor’s note**

**Spec: 3GPP TR 26.927 v0.8.0**

**Agenda item: 9.6**

**Document for: Agreement**

**1. Introduction**

This contribution updates the MPEG-FCM related work by adding more descriptive text and figures to the MPEG-FCM framework, adding also current performances achieved from the latest developments. It also updates the reference section with referenced MPEG documents.

The contribution also addresses the editor’s note on clause 6.3.4 about compression related function.

**2. Reason for Change**

Update MPEG developments in related work and solves the editor’s note

**3. Proposal**

It is proposed to agree and document the following changes to the 3GPP TR 26.927 v0.8.0.

\* \* \* first Change \* \* \* \*

# 2 References

[x1] Matsubara, Yoshitomo, Davide Callegaro, Sameer Singh, Marco Levorato, and Francesco Restuccia. "Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing." In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 337-346. IEEE, 2022.

\* \* \* End of first change \* \* \* \*

\* \* \* second Change \* \* \* \*

### 4.3.3 MPEG Feature Compression for Machines

MPEG related work, FCM (Feature Compression for Machines) addresses features compression. FCM is based on existing video compression standards. At the encoder, feature tensors are reduced, converted, and mapped onto packed video frames that may be encoded using encoders such as VVC, HEVC, or AVC, e.g., monochrome 10 bits video frames where the tensor channels are spatially packed. The video decoder outputs the packed video frames which are then processed to restore the feature tensors in their original shape, where the conversion, unpacking and feature restoration may use additional metadata transmitted along with the video bitstream.

Figure 4.3.3-1 shows a FCM encoder and decoder framework where the original selected trained model is split into two parts NN part1 and NN part 2. The current FCM framework under study includes a trained bottleneck, also called NN feature reduction at the encoder side and NN feature restoration at the decoder side. The parameters need to be trained for each split model and each split point, based on the end accuracy or the quality of reconstructed intermediate data after decoding, e.g. with an MSE-based metric. The NN feature reduction and restoration can be trained while keeping the parameters of the split AI model frozen. The FCM encoder and FCM decoder includes a Feature conversion function mapping features data onto packed video frames and vice versa.

The training of the feature reduction and feature restoration models may not require access to the original training set nor access to the same level of computational power. Current training strategies involve different training sets and using a loss based on metrics such as Mean Square Error to compare reconstructed features with the original intermediate features. Light trained codecs may be designed to fit existing large pre-trained split models without having to perform heavy computations involving the backpropagation of gradient from the final prediction of the tail of the model back to the original intermediate features. These strategies lead to the promising compression the performances detailed below.

The MPEG-FCM group is also considering untrained feature reduction methods that are computed online, enabling a more versatile codec that does not require retraining for each split model, but at the cost of lower compression efficiency.

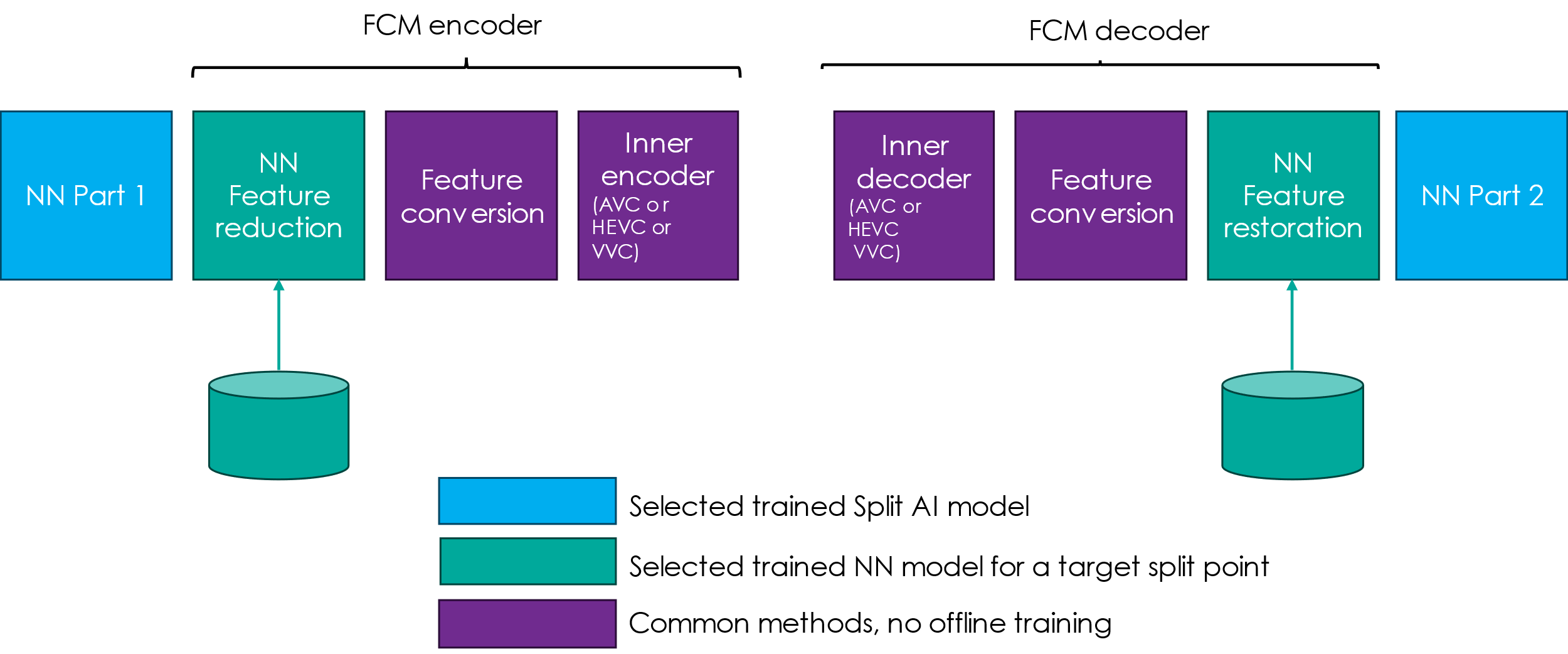


Figure 4.3.3-1: FCM framework

The current performance of FCM comparing to a remote inferencing anchor is 75% overall bitrate reduction over the same range of task performance as the remote inferencing anchor. The tasks include instance segmentation, object detection and object tracking. Remote inferencing refers to the compression of the input content using VVC reference software VTM-12.0 and the inference of the task model at the receiver on the decoded content. The compression ratio of the uncompressed feature size verses the compressed feature size in near lossless setting ranges from 6000:1 to 40000:1 on instance segmentation, object detection and object tracking. The obtained compression ratio of intermediate data while preserving near lossless accuracy is defined within a tolerance of 1% drop in task accuracy, relative to the performance achieved by the original task model operating directly on the input data.

\* \* \* End of second change \* \* \* \*

\* \* \* Third Change \* \* \* \*

## 6.3 Intermediate data

### 6.3.1 Introduction

### 6.3.2 Intermediate data size delivery

### 6.3.4 Compression related functions

Depending on the AI media service use case (and the required AI task) some compression approaches (e.g., quantization, entropy coding, transformations) can be used to reduce the size of the transferred intermediate data and to adapt the split AI/ML operations between the UE and the network to changing conditions.

Compression functions such as quantization, entropy coding, pruning may be applicable to any intermediate data tensors. Some of these functions require corresponding decompression processes to decode and readapt the intermediate data for the inference of the second part of the model. Different ratios of reduction of intermediate data size can be reached for each split point configuration, for instance by varying quantization parameters. These agnostic compression functions can be used for any model, any split point, any type of model task with different input media data (image, video, audio, text). Agnostic compression evaluations with on-the shelf compression functions provide promising performances in terms of intermediate data size relative to the accuracy of the results

Another approach to further compress intermediate data is to inject a so-called bottleneck to create smaller intermediate data with an auto-encoder-like structure. Two main cases can be distinguished.

* An original design of the split model can include natural split points, i.e. intermediate layers where the dimensions of intermediate output are reduced by design and their entropy is controlled. The training of this model can be done using a loss function which includes both the end accuracy and the estimation of the size of the bitstream of intermediate data. [x3] shows an example of intermediate data bottlenecks using an embedded autoencoder.
* A compression function including bottleneck can be inserted at the encoder side and a corresponding decompression function at the decoder side without modifying the original structure of the split model. The model including the bottleneck may need to be designed and trained for each split model and each split point. Envisioned frameworks may enable a sender to select an optimised function from a set of adaptive compression/decompression functions based on different trained bottlenecks applying different compression factors. The encoder may need to send an indication of the compression profile used for the intermediate data transfer to the decoder side to select and use the corresponding decompression function. For instance, MPEG-FCM as mentioned in clause 4.3 is working on the standardization of such codec, with a current model that includes both the bottleneck concept with trained feature reduction at the encoder and restoration at the decoder. In addition to the feature reduction, the current design maps resulting reduced feature tensors to video frames in order to utilize existing video codecs such as H.264 AVC, H.265/HEVC or H.266/VVC to further compress the features and take advantage of advance temporal compression methods in the case of feature streaming. MPEG-FCM currently delivers very promising performances on intermediate data compression applied to video making use of video decoders such as HEVC as shown in table 6-3.4.1

The table 6.3.4-1 below summarizes the different approaches and characteristics under consideration by MPEG. The reported compression ratios consider the size of the resulting bitstreams vs. the original dumped files of intermediate features coded on 32 bits, at near lossless accuracy, defined as a final accuracy drop of less than 1% of the original (un-split) model. The compression ratios highly depend on the split model and target task. For instance, for MPEG-FCM, ratios range from 6000:1 in the case of still image object detection using a split Faster-RCNN model to 40000:1 in the case of video object tracking.

Table 6.3.4-1: Approaches and characteristics .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Approaches | Agnostic | Training required | Number of split points | Compression ratio | Reference |
| Bottleneck model | No | Yes | One to several | Unknown | research [x1] |
| Basic quantization | Yes | No | Any | 2:1 to 4:1 | 26.847 |
| Quantization with Entropy coding with NNC (Neural Network Coding) Codec | Yes | No | Any | 5:1 to 10:1 | 26.847 |
| MPEG-FCM (current) | No | Yes (\*) | One | 6000:1 to 40000:1 | TBD |

(\*) A retraining is only required for the additional MPEG-FCM functions (see clause 4.3.3).

\* \* \* End of third change \* \* \* \*