**Source:** **InterDigital Belgium. LLC Title: [FS\_AI4Media] pCR Split operations on AI/ML models**

**Spec: 3GPP TR 26.927 v0.7.0**

**Agenda item: 9.6**

**Document for: Agreement**

1. **Introduction**

Split inferencing involves splitting a selected model into several subsets, with one subset inferred in a first endpoint and another in a second endpoint.

A common model representation used for existing formats or frameworks described in the TR, such as ONNX, NNEF, TensorFlow and PyTorch, is a computational graph or directed acyclic graph.

The contribution presents generic split elements on applying a split operation to directed acyclic graphs. It explains the split operation applied to simple or complex model graph such as multi-branch split. It introduces information on how to split a model using the de-facto standard ONNX model format, which is widely used for almost all models.

1. **Reason for Change**

Currently, nothing is documented on split AI/ML models, which makes the current TR incomplete.

1. **Proposal**

We proposed to agree the following changes to 3GPP TR 26.927 v0.7.0.

We suggest moving clause 6.4 on Existing frameworks for AIML as 6.2 before Model data and Intermediate data clauses for introducing generic aspects before.

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

## 6.3 Intermediate data

### 6.3.1 Introduction

Split AI/ML operation is defined as the distribution of AI/ML model inferencing between at least two endpoints, for example a UE and a Network endpoint. The data output from the first endpoint (intermediate data) is delivered to the second endpoint to guarantee the expected user experience on running a particular AIML application regarding UE, Network and server capabilities. Requirements for such a split inference service may include avoiding service interruption, and optimizing the network, UE or server resources.

### 6.3.2 Intermediate data size delivery

Intermediate data characteristics depend on various aspects described in clause 4.1 and clause 4.5 including intermediate data volume or size.

Different factors can impact the size of the intermediate data for delivery, which may require the adaptation of split AI/ML operations between the UE and the network:

* AI inference task use-case and requirement: The service requirements on an AI task drives the intermediate data size. For example, a complex AI task for detecting multiple objects in a dense and moving video requires far more intermediate data than for a simpler AI task on static scene or about a single object.
* AI model for the AI inference task: Different trained AI models for the same AI inference task can be available with different characteristics on not only the AI model architecture and size, but also on the intermediate output size, depending on the split point(s).
* Split point selection: The selection of a split point within an AI model determines the dimension of the intermediate data. The output size at a given split point compared to another may vary from 1 to 5 or more [aa].
* Adapted trained model for split operation: Adapted models can be designed to provide reduced intermediate data at identified split points [cc].
* Optimization: accuracy/quality metrics determine the result of a split inference. Basic precision quantization, from 32 bits to 16/8 bits may reduce the overall size of intermediate data while still meeting the required output result quality/accuracy for the service.
* Inference input video frame rate adjustment: The input frame rate in case of video determines the streaming bitrate of the intermediate data to be delivered. An AI inference task may not produce media content and does not necessarily need to produce an output result at 30 or 60 frames as in the case of video streaming.
* Non-real time delivery: The transmission of intermediate data may not necessarily need to be delivered in a real-time based manner. The result of inferencing split model on an image, a set of images or a video sequence may not require an immediate result. The transmission of intermediate data can be done progressively with a constrained bandwidth,
* Different input image resolutions may produce different intermediate data size for models with variable input size (e.g. image classification models)

### 6.3.3 Operations for splitting AI/ML models

For split configurations, clause 5.1, an AI/ML model is split in two subsets consisting of a head subset or a part 1 running on a first endpoint (UE resp. Network) and a tail subset or part 2 running on a second endpoint (Network resp. UE). The first endpoint provides intermediate data to the second endpoint over the network.

The computational graph of a model above can be represented by a directed acyclic graph with existing formats or frameworks such as ONNX, NNEF, TensorFlow and PyTorch as described in clause 6.2.5 and 6.4.

A simple split operation on a particular node in such a graph consists of a single branch split when the split node has only one input edge and if the split occurs before the split node (resp. one single output edge if the split occurs after the split node).

A multi-branch split occurs when a split node has more than one input and if the split occurs before the node or more than one output if the split occurs after the node. Certain specific aspects may need to be considered when applying a split operation to divide a model into two or more subsets:

* A model graph can start taking input data into account at a node later than the first node. In this case, if the split occurs before the node requiring the input data, this input data needs to be provided to the second part of the subset running in the second endpoint, in addition to the intermediate data generated by the first part.
* A model graph can comprise multiple nodes producing partial results. In this case, a split occurring after a node producing partial results will require to collect partial results from both endpoints to compute the final consolidated results.

ONNX is an interoperable format available for frameworks 6.4 and for a large number of models ( <https://github>.com/onnx/models). Splitting a single branch is straightforward using *extract\_model* function (<https://onnx.ai/onnx/api/utils.html>). A generic multi-branches split using the same function can also be performed after parsing the model to obtain the list of inputs and outputs required by each model subset.

Different endpoints can split an ONNX model down to the split node or from the split node to the end by using the same node identification and parsing rules to apply to the graph. For example, a first endpoint can build the head subset from the full ONNX model and the second endpoint can build the tail subset from the same full ONNX model.

One endpoint can split model an ONNX model file in two subsets and deliver the required subset to an endpoint

For dynamic split point reselection, an endpoint may prepare different model subsets for the different candidate split point configurations or build the split model subset ‘on demand’ from a full ONNX model when it is required.

\* \* \* End of first Changes \* \* \* \*