**Source: Interdigital Finland Oy**

**Title: [FS\_AI4Media] pCR on update metadata for split operations**

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

**Agenda item: 9.6**

**Document for: Agreement**

1. **Introduction**

The contribution mainly updates clause §6.6.4 “Intermediate data information for split AI/ML operations” of TR 26.927 v0.8.0. It refines intermediate data as a composition of a list of tensors data instead of a single tensor. It addresses the multi-branch split use-cases generating intermediate data as a list of tensors as it has been demonstrated in several splitting evaluations presented in SA4 #128. We update compression algorithms which can be global to all intermediate data tensors or specific to each intermediate data tensor from the list.

The second change is an update to the section numbers solving numbering of two sections with 6.6.3.

Finally, we propose to remove the editor’s note of clause 6.6 metadata.

1. **Reason for Change**

Update the metadata section of the TR to address multi-branch split use-case.

1. **Proposal**

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

\* \* \* beginning of Change \* \* \* \*

## 6.6 Metadata

### 6.6.1 Introduction

Metadata for AI media services may include information describing AI models, inference requirements, endpoint capabilities (UE or network) and information more specific to the configuration, control and management of the basic AI service scenarios (AI model delivery, split AI/ML operation and distributed/federated learning).

NOTE: The delivery of the metadata described in this clause is not specified.

NOTE 2: How consistent and interoperable signalling of metadata (such as of existing framework mentioned in clause 6.4) can be ensured is FFS.

### 6.6.2 Common AI model information

AI model information metadata is used to describe the characteristics of AI models which may be used for an AI media service. This information may be common to all three AI service scenarios, and may be used in the selection of a suitable AI model by the UE or network, given an AI media service.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metadata category** | **Metadata type** | **Definition** | **Metadata type description (Examples)** |
| **Model information** | **Model identifier** | An identifier for an AI model (or variants of it) specified for a certain AI media service. The identifier may be a name, a number, a combination thereof, a hash value. The identifier is defined during the configuration stage. | model\_1, model\_2 |
| **Number of parameters** | Total number of parameters in the neural network. | 11 million |
| **Model size** | The size of the AI model file in megabytes. | 40MB |
| **Input size** | The maximum size of the input data supported by the AI model in kilobytes. | 256 KB |
| **Output size** | The maximum size of the output data supported by the AI model in kilobytes. | 256 KB |
| **Accuracy** | The trained accuracy of the AI model as a percentage. | 85% |
| **Target inference latency** | The target inference latency specified for a given AI model in milliseconds. Such latency is measured between the input and output layers of the AI model at inference. This value is related to the service inference latency requirement of the service for which the AI model is provided, as well as the typical hardware capabilities of an entity performing the inference of the model. | 20ms |
| **Format/ framework** | The format or framework used to express the AI model, including its version number. | Pytorch 2.0 ONNX 1.15.0 |
| **Processing capabilities** | Estimated capabilities for processing the model including the computational power such as the computational cost (in FLOPS), the computational complexity (in MAC operations). It also includes the temporary memory to store model parameters.  | NPU 10TFLOPS, MEM 10GB |

### 6.6.3 AI model information for split AI/ML operations

AI model information metadata for split AI/ML operations is used to describe the characteristics of AI models for split inference service scenarios. This information may be used in the selection of a split point (from which a multiple may be predefined by the service provider for a certain AI media service). A trained model can be represented as a directed acyclic graph model represented by a collection of nodes interconnected with edges (e.g. ONNX). A split point may happened before or after a graph node identified by its name or a number.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metadata category** | **Metadata type** | **Definition** | **Metadata type description (Examples)** |
| **Split model information** | **Split points** | The number of predefined split points at which a certain model can be divided into two for split inferencing. | 2 |
| **Split point information** | **Split point identifier** | An identifier of the split point in a description of a computing graph, may be generated by a neural network description language such as ONNX/NNEF. Identifiers must guarantee unique identification of a specific split point. | Nb:10, 75 Name: Layer\_10, |
| **Split point intermediate data size** | The size of the intermediate data resulting from the given split point, in kilobytes. Intermediate data size is typically dependent on the tensor size at the given split point. | 1086KB |
|  | **Split point number** | The number of the split point where the split occurs. The number may belong to set of identified numbers defined at the configuration stage.  | 10 |
| **Split point name** | The name of the split point where the split occurs. The name may belong to set of identified split point names defined at the configuration stage.  | conv2d\_1234 |
| **Split point flag** | An information on whether to consider the split point before the split point identifier or after. The convention on whether it is before or after may be defined at the configuration stage. | before, after |

### 6.6.4 Intermediate data information for split AI/ML operations

Intermediate data information identifies the structure of intermediate data output from a first endpoint that need to be retrieved to feed the inference of the second endpoint after transmission of the intermediate data over the network.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metadata category** | **Metadata type** | **Definition** | **Metadata type description (Examples)** |
| **Intermediate data general information** | Tensor structure framework | The exact underlying tensor structure of the intermediate data tensors including the exact version of it. | PyTorch 2.0,TensorFlow v2.13.0, NumPy v1 .25 |
| ~~Tensor shape~~ | ~~The tensor shape(s) when the output is intermediate data. Tensor shape is a tuple of positive integers, where the size of the tuple represents the dimension of the tensor, and each value represents the size in each dimension.~~  | ~~[1,64,64,64].~~  |
| ~~Tensor element data type~~ | ~~The data type of each output intermediate data tensor~~ | ~~:int64, Float32~~ |
| Data direction  | This defines the direction of transmitted data, either uplink (from UE endpoint to network endpoint) or downlink (From a network endpoint to the UE endpoint). This information may be useful to configure an intermediate data delivery session | Upstream, Downstream |
| Global compression algorithm | Identifies a compression algorithm that can be applied to all the intermediate data tensors. For example, when the connectivity condition between the UE and the network is insufficient to transmit the original intermediate data, a compression algorithm may be applied. | NONE, FCM, SNAPPY, … |
| **Intermediate data tensor information** | Tensor list | List of Tensors that composed the intermediate data | [tensor1, tensor2, tensor3, tensor4] |
|  |  | Tensor name | The name of the tensor | Tensor1  |
|  |  | Tensor shape | Tensor shape is a tuple of positive integers, where the size of the tuple represents the dimension of the tensor, and each value represents the size in each dimension.  | [1,64,64,64].  |
|  |  | Tensor data type | The data type of each intermediate data tensor | Float32, int32 |
|  |  | Tensor compression algorithm | Identifies the compression algorithm(s) that can be applied to a particular tensor. The tensor compression algorithm supersedes the global compression algorithm when both are defined | NONE, FCM, SNAPPY, … |

### 6.6.5 Service requirement information

Service requirement information metadata is used to describe the latency and processing requirements for the AI media service. Such information may be used in the selection of an AI model for the service, and/or the selection of a split point for a certain AI model for split inferencing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metadata category** | **Metadata type** | **Definition** | **Metadata type description (Examples)** |
| **Service requirement information** | **Maximum service inference latency** | The maximum inference latency requirement specified for a given AI media service, in milliseconds. In the case of split inferencing, this requirement includes the delivery latency of the intermediate data between the first and second split inference entities. | 100ms |
| **Minimum service inference accuracy** | The minimum accuracy specified for a given AI media service. | 80% |
| **Service type identifier** | An identifier for the service type to be supported by the AI/ML model, such as ASR (Automatic Speech Recognition), TTS (Text To Speech), Translation (with the indication of input and output languages). | TTS, ASR, Trans-EN-to-ZH |
| **Service accuracy**  | The expected service accuracy  | 85% |

### 6.6.6 Endpoint capability information

The endpoint capability information includes the capabilities of the endpoint (UE or network) for processing and transmitting the AI/ML model and intermediate data. Such information can be updated due to the change of the endpoint’s work load or the network conditions. It can be used for the selection of AI model, split inference, intermediate data compression, progressive model delivery.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metadata category** | **Metadata type** | **Definition** | **Metadata type description (Examples)** |
| **Endpoint capability information** | **Processing capabilities** | The available resources for processing AI/ML model including the computational power (in FLOPS), the memory to store model parameters and perform the inference. | NPU 10TFLOPS, MEM 10GB |
| **Supported AI Framework** | The AI framework(s) supported by the endpoint.  | TensorFlow 2.0 |
| **Supported compression algorithms** | The supported compression algorithm(s) for intermediate data compression. | NONE, FCM, SNAPPY, … |
| **Connection capabilities** | This indicates the available bandwidth in bit/s between the UE and the network for transmitting the AI model and/or the intermediate data. | 256 kb/s |

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