**3GPP TSG SA4 123-e MeetingS4-230565**

**Online, 17th – 21th April 2023**

**Source: Fraunhofer Heinrich Hertz Institute (HHI)**

**Title: [FS\_AI4Media] Scenario for transmission of AI/ML model data**

**Agenda item: 9.7**

**Document for: Discussion and Agreement**

1. Introduction

At the 122th SA4 meeting in Athens, it was agreed to define an AI/ML evaluation framework, including a set of anchor models and corresponding data sets. Then, at the SA4 Video SWG Teleconference on March 28th, it was decided to first define scenarios and then to derive the test framework from these scenarios. This document suggests such a scenario.

The scenario covers the transmission of AI/ML model data, which is required in several use cases defined in clause 4 of the permanent document [1] and has been identified as one of key operations for AI/ML related services in TR 22.874 [2]. As staring point, we suggest a test case for this scenario considering a state-of-the-art model (*wav2vec*) for automatic speech recognition (ASR). Further test cases might be added in future meetings.

R2 of the document includes changes triggered by the e-mail discussion on April 18th and minor typo fixes.

1. Proposed Changes to clause 7 (AI/ML evaluation framework)
2. AI/ML evaluation framework

# Scenario: Transmission of compressed AI/ML model data for automatic speech recognition

## Motivation

AI/ML model data distribution and sharing over 5G system has been identified in TR 22.874 [2] as one of the three key operations for AI/ML related services. Reason for this is that UEs might need a great variety of AI/ML models to respond to different tasks and environments, while not being able to store all needed AI/ML models due to memory storage constraints, so that a frequent context adaptive down-loading of AI/ML model data is necessary.

To tackle this problem, methods for model compression have been proposed (see clause 6), which provide the benefits that they 1) lower bandwidth requirements or latencies for model data distribution, and 2) reduce the memory footprint of the AI/ML models on the UEs. However, besides the reduction of the model size, compression method can also lead to a decrease of the AI/ML model performance. Which performance/compression trade-offs can be reached by different AI/ML model compression methods is thus an important question when defining AI/ML related services and is thus investigated in this scenario.

## Test Cases

From the media-based AI/ML use cases defined in clause 4, the following require the transmission of AI/ML model data and thus could benefit from model compression:

1. Full or partial transfer of models for object recognition in image and video (clause 4.1)
2. Transfer of models for post-filtering for video coding (clause 4.2.1.2)
3. Transfer of models for crowd-sourcing media capture (clause 4.3.1)
4. Transfer of models for NLP on speech (clause 4.4)

Based on these use cases, different test cases are defined in the following.
[Note: More test cases might be added in future.]

### Model for Automatic Speech Recognition

This test case evaluates the transmission of the *wav2vec* AI/ML model [3] for automatic speech recognition (ASR), which derives a transcript of a given speech sequence.

The transmission of compressed AI/ML models for ASR is relevant in the following use cases defined in clause 4:

* Crowd-Sourcing Media Capture (clause 4.3.1): To adapt to background noise or for lyrics recognition, specialized AI/ML models for ASR need to be transferred to a huge number of UEs for device inference.
* NLP on Speech (clause 4.4.): An initial ASR model needs to be down-loaded to the UE; then updated model data needs to be shared frequently with other UEs for distributed/federated learning.

How the *wav2vec* AI/ML model can be employed by an UE is shown in Figure 7.1.2.1-1, which comprises the following entities:

* A speech sequence stored as uncompressed audio file sampled with 16kHz.
* The *wav2vec* AI/ML model inferring a classification for the speech sequence.
* A vector sequence representing the classification. Each vector comprises 29 elements specifying the probability (represented as logits) of the 29 labels: '*-*', ' ', 'E', 'T', 'A', 'O', 'N', 'I', 'H', 'S', 'R', 'D', 'L', 'U', 'M', 'W', 'C', 'F', 'G', 'Y', 'P', 'B', 'V', 'K', ''', 'X', 'J', 'Q', and 'Z'.
* A label selector selecting the most probable labels from the vector sequence.
* The predicted transcript, i.e. the sequence of selected labels.



**Figure 7.1.2.1-1: Prediction of a transcript with the *wav2vec* AI/ML model**

* + - * 1. Test Setup

The purpose of the test setup is to characterize different compression methods to compare their performance when compressing the *wav2vec* model. The test setup to derive metrics for a given method under test is shown in Figure 7.1.2.1.1-1. It comprises of the following entities:

* A test encoder, which can also be a sender-only optimization/compression technique, and (optionally) a test decoder implementing the method under test.
* A test configuration for the test encoder.
* A reference model that is compressed by the test encoder.
* A test bitstream representing the compressed reference model.
* A reconstructed model that is either a) reconstructed by the test decoder from the test bitstream or b) equal to the test bitstream for encoder-only compression methods.
* Metrics computation based on the reference model, the test bitstream and the reconstructed model.
* The derived test metrics for the method under test with respect to the reference model and the test configuration.



**Figure 7.1.2.1.1-1: Derivation of test metrics for a method under test implemented by the test encoder and decoder**

The test metrics are defined in clause 7.1.2.1.2. How they characterize a method under test is defined in clause 7.1.2.1.3. Clause 7.1.2.1.4 provides further information about the model and test data and an exemplary script implementing the metric computation.

* + - * 1. Metrics

The reference model and test bitstream are provided as files containing the model parameters. The file size (*size*) combined with the achieved word error rate (*wer*) are employed to determine the efficiency of a compression method.

**File Size**

The reference model and tested model can be stored as follows:

1. The reference model is provided as data file containing *numParam* uncompressed model parameters individually represented as *N*-byte floating-point values.
2. For encoder-only compression methods, the test bitstream is provided as data file containing *numParam* quantized and/or reduced model parameters individually represented as *N*-byte values.
3. For methods requiring a decoder, the test bitstream is a coded representation encoding the parameters jointly.

For all cases, *size* can be determined by measuring the file size. For cases a) and b), *size* can also be determined as *numParam* \* 8 \* *N*.

**Word Error Rate**

To quantify the performance of the reference and the reconstructed model, the word error rate (*wer*) is used, which has also been applied in the original publication of the *wav2vec* model [3]. The word error rate is determined based on the Librispeech *test-clean* data set [6][7], which contains 2620 data pairs. Each pair comprises

* a speech sequence stored as uncompressed audio file, and
* a reference transcript of the audio sequence stored as text file.

Using the data set, the *wer* value is determined in two steps:

1. A word error rate $wer\_{i}$ is derived for each pair $i$ of the test set as follows:
* The AI/ML model is applied as shown in Figure 7.1.2.1-1 using the speech sequence as input and obtaining a predicted transcript as output.
* The predicted and reference transcripts are split into a predicted and a reference list of words, respectively.
* The word error rate $wer\_{i}$ of the predicted word list with respect to the reference word list is derived as follows

$$wer\_{i}=\frac{S\_{i}+D\_{i}+I\_{i}}{N\_{i}} $$

with $S\_{i}$, $D\_{i}$, and $I\_{i}$ denoting the number of word substitutions, word deletions, and word insertions in the predicted word list and $N\_{i}$ denoting the number of words in the reference list.

1. The total word error rate *wer* is derived as follows:

$$wer= \frac{\sum\_{i}^{}wer\_{i}N\_{i}}{\sum\_{i}^{}N\_{i}}$$

* + - * 1. Characterization

To characterize a compression method under test, it is evaluated using different test configurations *T*, which might be produced by varying encoder parameters as e.g. quantization parameters or sparsification ratios. The characterization is performed in comparison to the model size *sizeRef* and the word error rate *werRef* of the reference model, which are given in Table 7.1.2.1.3-1.

|  |  |  |
| --- | --- | --- |
| *numParam* [M] | *sizeRef* [Mbit] | *werRef* [%] |
| 94.393245 | 3020.583840 | 3.397 |

**Table 7.1.2.1.3-1: Number of parameters, size and word error rate of the *wav2vec* reference model**

To fully characterize a compression method under test, a data pair (*cSize, wer*) is derived for each test configuration *T* from a set of test configurations with

* *cSize* denoting the size of the test bitstream *size* divided by the size of reference model *sizeRef* and
* *wer* denoting the word error rate of the reconstructed model.

The set of test configurations should at least contain 5 test configurations *T* that produce word error rates in the range of *werRef* (3.397%) to *werRef*+ 0.05 (8.397%). [Note: This threshold might be discussed and changed.]

For comparison, (*cSize, wer)* pairs, as well as *werRef,* might be reported graphically, as shown in Figure 7.1.2.1.3-1.

**

**Figure 7.1.2.1.3-1: Example for the characterization of a method *M* for different test configurations *T***

* + - * 1. Model Data, Test Data, and Exemplary Script

Several pre-trained *wav2vec* AI/ML models are provided by the TorchAudio library [4] under MIT License. For evaluation, the *BASE* *LS-960* *wav2vec* model, as defined in [3], should be used. It is trained on 960 hours of audio data from the Librispeech data set [6][7] and is called *WAV2VEC2\_ASR\_BASE\_960H* [5] in the TorchAudio library.

For evaluation, the Librispeech *test-clean* data set [6][7] should be used, which is available under Creative Commons Attribution 4.0 International License.

[Note: Alternatively or additionally the following pre-trained models and data sets might be used:

* Model WAV2VEC2\_ASR\_LARGE\_960H: Lower *wer*, but larger size.
* Test set Librispeech *test-other*: More challenging test data.]

An exemplary python-script to derive word error rate and file size of the *wav2vec* model is given in Figure 7.1.2.1.4-1.

**import** torchaudio
**import** torch
**import** torchaudio.datasets **as** datasets
**from** torcheval.metrics **import** WordErrorRate

test\_dir = **"D:\\data"** *# This directory should exist.*device = **"cuda"

def** main():
 *####### Get Model ##############################* bundle = torchaudio.pipelines.WAV2VEC2\_ASR\_BASE\_960H
 model = bundle.get\_model()
 sample\_rate = bundle.sample\_rate
 labels = bundle.get\_labels()

 *####### Get Data Loader Model ##################* val\_loader = torch.utils.data.DataLoader(
 datasets.LIBRISPEECH(test\_dir, **"test-clean"**, **"LibriSpeech"**, **True** ),
 batch\_size=1, shuffle=**False**,
 num\_workers=1, pin\_memory=**True**)

 *####### Evaluate Model #########################* model.eval()
 model.to( device )
 metric = WordErrorRate()
 blank = 0

 **with** torch.inference\_mode():
 **for** speech\_sequence, cur\_sample\_rate, reference\_transcript, \*dump **in** val\_loader:
 *# Resample speech sequence if necessary* **if** cur\_sample\_rate != sample\_rate:
 speech\_sequence = torchaudio.functional.resample(speech\_sequence, cur\_sample\_rate, sample\_rate)
 speech\_sequence = speech\_sequence.reshape( (1,-1) )
 speech\_sequence = speech\_sequence.to(device)

 *# Apply wav2vec* vetor\_sequence, \_ = model(speech\_sequence)

 *# Select labels* idcs = torch.argmax(vetor\_sequence[0], dim=-1)
 idcs = torch.unique\_consecutive(idcs, dim=-1)
 idcs = [i **for** i **in** idcs **if** i != blank]
 predicted\_transcript = **""**.join([labels[i] **for** i **in** idcs])
 predicted\_transcript = predicted\_transcript.replace(**"|"**,**" "**)

 *# Update error* metric.update( predicted\_transcript, reference\_transcript[0] )

 wer\_Ref = metric.compute()
 print(**'wer\_Ref: {wer\_Ref:.3f} %'**.format(wer\_Ref=wer\_Ref\*100))

 *####### Get Model Size #########################* num\_parameters = 0
 **for** param **in** model.parameters():
 num\_parameters += param.numel()

size\_Ref = num\_parameters \* 4 \* 8 *# Each parameter is stored as 32 bit float*
 print(**'size\_Ref: {size\_Ref:.3f} Mbit'**.format(size\_Ref=size\_Ref/1000/1000))

**if** \_\_name\_\_ == **'\_\_main\_\_'**:
 main( )

**Figure 7.1.2.1.4-1: Exemplary python script for determining *sizeRef* and *werRef***

1. References
2. S4-230305 [FS\_AI4Media] Permanent Document v0.6, February 2022.
3. 3GPP TR 22.874, Study on traffic characteristics and performance requirements for AI/ML model transfer in 5GS
4. A. Baevski, H. Zhou, A. Mohamed and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”, arXiv, 2006.11477, 2020
5. TorchAudio: An audio library for Pytorch [Computer software], <https://github.com/pytorch/audio>, V0.13.1
6. TorchAudio: WAV2VEC2\_ASR\_BASE\_960H, [Computer software] https://pytorch.org/audio/stable/generated/torchaudio.pipelines.WAV2VEC2\_ASR\_BASE\_960H.html
7. OpenSLR, LibriSpeech ASR corpus [Online], <https://www.openslr.org/12>
8. V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Australia, 2015