**Source**: Interdigital Finland Oy

**Title:** [FS\_AI4Media] Frameworks for evaluation

**Agenda item:** 9.7

**Document for:** Discussion and Agreement

1. **Introduction**

It was agreed at the SA4-122 meeting to select one or several frameworks and/or libraries and a list of models as a baseline to conduct the traffic characteristics measurement on AIML data.

This contribution discusses candidate frameworks/libraries and details the characteristics of the most popular frameworks, namely TensorFlow/Keras and PyTorch.

1. **proposed changes.**

--------------------------------------------- Begin change -------------------------------------------------------------------------

# 7 AI/ML evaluation framework

## 7.X AI Framework/Libraries

An AI/ML framework brings a set of services which are interfaces, libraries or tools. They are used to create models, train them and/or to infer input data and deliver a prediction.

Hereafter is a short list:

1. TensorFlow
2. PyTorch
3. Caffe
4. Keras
5. MXNET
6. Darknet

Some frameworks are especially designed for on-device (Mobile Phones) deep Learning, we may present the two main ones:

1. TensorFlow Lite [1]
2. PyTorch Live [2]

**Note**: Keras is running on top of TensorFlow, and both together provide a high-level APIs to make a more user-friendly framework. For the rest of the document TensorFlow and Keras frameworks are considered as one entity noted TensorFlow/Keras.

AI/ML frameworks can be completed and enriched with libraries, for example to provide optimization and compression tools such as:

* NNC : clause §6.5.7
* AI Model Efficiency ToolKit (AIMET) clause §6.6.

Both libraries support TensorFlow/Keras and PyTorch environments.

### 7.X.1 Framework popularity

PyTorch and Tensorflow/Keras are the two major and most popular frameworks for Deep Learning.

PyTorch appears significantly more in academics as shown in the next graph [4]



On the other hand, Tensorflow is much more popular in industry.

The TensorFlow eco-system comprises some deployment-oriented applications like TensorFlow Serving and TensorFlow Lite for AI/ML application to be deployed on cloud, edge, server, mobile or IoT devices.

PyTorch has filled the gap by proposing TorchServe [5] and PyTorch Live [2].

### 7.X.2 Detailed frameworks characteristics

**Framework/Libraries tools available (compression, quantization,)**

TensorFlow and Pytorch natively support optimization and quantization tools.

**Hardware accelerator support**

List of tools for optimizing the ML models.

It is very likely that the model performance will be evaluated with various processing conditions being CPU, GPU, TPU or others like DSP.

TensorFlow/Keras and PyTorch already integrate such capabilities:

TensorFlow/Keras GPU or TPU usage in respectively [6] and [7]

PyTorch GPU or TPU usage in respectively [8] and [9]

**Supported models.**

Natively both frameworks TensorFlow/Keras and PyTorch integrate many pre-trained models, this is described in document “models for evaluation”. If the model is not available, it can be reconstructed from its known architecture and trained.

A list of pre-trained model support is proposed for keras in [10], and for Pytorch in [11].

**Split function**

Splitting functionality shall be evaluated to point out the benefits it can bring to the 5G system (latency, energy, privacy), but also to measure and characterize the intermediate data. Therefore, the framework shall offer APIs/functions to split some models. This function is already available in TensorFlow/Keras framework as described in doc Split scenarios for evaluation and TensorFlow based split evaluation platform.

**Mobile or on-device version**

Both PyTorch and TensorFlow/Keras have their own mobile solutions TensorFlow Lite [1] and PyTorch Live [2].

**Language**

Both PyTorch and TensorFlow/Keras are Python based.

TensorFlow supports additionally JavaScript, C++ and Java.

**Supported format for AI/ML model**

PyTorch and TensorFlow/Keras support Open Neural Network eXchange (ONNX) and Neural Network Exchange Format (NNEF).

* **ONNX** : Tensorflow models (including Keras and TensorFlow Lite models) can be converted to ONNX [12]. PyTorch models can be exported to the ONNX format [13]. ONNX support tools for porting PyTorch model into TensorFlow or vice-versa.
* **NNEF**: supported by Khronos and designed to support both PyTorch and TensorFlow. NNEF tools can convert trained models from/to ONNX format [14].

### 7.X.3 References

1. <https://www.tensorflow.org/lite>
2. <https://playtorch.dev/>
3. <https://github.com/quic/aimet>
4. <https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/>
5. <https://pytorch.org/serve/?ref=assemblyai.com>
6. <https://www.tensorflow.org/guide/gpu>
7. <https://www.tensorflow.org/guide/tpu>
8. https://pytorch.org/docs/stable/notes/cuda.html /GPU support through CUDA
9. https://pytorch.org/xla/release/2.0/index.html XLA/TPU
10. <https://modelzoo.co/framework/keras>
11. <https://modelzoo.co/framework/pytorch>
12. <https://onnxruntime.ai/docs/tutorials/tf-get-started.html>
13. <https://pytorch.org/docs/stable/onnx.html>
14. <https://www.khronos.org/api/nnef>

--------------------------------------------- end of change -------------------------------------------------------------------------

1. **Proposal**

We propose to include the text in the clause 7 of the permanent document as a baseline for refinements and improvements.