The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP.  
The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.  
This Report is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.  
Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices.

3GPP TR 22.874 V18.2.0 (2021-12)

Technical Report

3rd Generation Partnership Project;

Technical Specification Group Services and System Aspects;

Study on traffic characteristics and performance requirements for AI/ML model transfer in 5GS

(Release 18)



Keywords

AMMT, AI, ML

***3GPP***

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis

Valbonne - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© 2021, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).

All rights reserved.

UMTS™ is a Trade Mark of ETSI registered for the benefit of its members

3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  
LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners

GSM® and the GSM logo are registered and owned by the GSM Association

Contents

Foreword 6

Introduction 6

1 Scope 7

2 References 7

3 Definitions, symbols and abbreviations 10

3.1 Definitions 10

3.2 Abbreviations 10

4 Overview 11

5 Split AI/ML operation between AI/ML endpoints 12

5.1 Split AI/ML image recognition 12

5.1.1 Description 12

5.1.2 Pre-conditions 14

5.1.3 Service Flows 14

5.1.4 Post-conditions 15

5.1.5 Existing features partly or fully covering the use case functionality 15

5.1.6 Potential New Requirements needed to support the use case 15

5.1.6.1 Potential KPI Requirements 15

5.2 Enhanced media recognition: Deep Learning Based Vision Applications 16

5.2.1 Description 16

5.2.2 Pre-conditions 17

5.2.3 Service Flows 17

5.2.4 Post-conditions 17

5.2.5 Existing features partly or fully covering the use case functionality 17

5.2.6 Potential New Requirements needed to support the use case 18

5.3 Media quality enhancement: Video streaming upgrade 18

5.3.1 Description 18

5.3.2 Pre-conditions 19

5.3.3 Service Flows 19

5.3.4 Post-conditions 19

5.3.5 Existing features partly or fully covering the use case functionality 19

5.3.6 Potential New Requirements needed to support the use case 19

5.4 Split control for robotics 20

5.4.1 Description 20

5.4.2 Pre-conditions 21

5.4.3 Service Flows 21

5.4.4 Post-conditions 22

5.4.5 Existing features partly or fully covering the use case functionality 22

5.4.6 Potential New Requirements needed to support the use case 22

5.4.6.1 Potential KPI Requirements 22

5.5 Session-specific model transfer split computation operations 23

5.5.1 Description 23

5.5.2 Pre-conditions 23

5.5.3 Service Flows 23

5.5.4 Post-conditions 24

5.5.5 Existing features partly or fully covering the use case functionality 24

5.5.6 Potential New Requirements needed to support the use case 25

6 AI/ML model/data distribution and sharing over 5G system 25

6.1 AI/ML model distribution for image recognition 25

6.1.1 Description 25

6.1.2 Pre-conditions 26

6.1.3 Service Flows 26

6.1.4 Post-conditions 27

6.1.5 Existing features partly or fully covering the use case functionality 27

6.1.6 Potential New Requirements needed to support the use case 27

6.1.6.1 Potential KPI Requirements 27

6.2 Real time media editing with on-board AI inference 28

6.2.1 Description 28

6.2.2 Pre-conditions 29

6.2.3 Service Flows 30

6.2.4 Post-conditions 30

6.2.5 Existing features partly or fully covering the use case functionality 30

6.2.6 Potential New Requirements needed to support the use case 30

6.2.6.1 Introduction 30

6.2.6.2 Potential KPI Requirements 32

6.3 AI/ML model distribution for speech recognition 33

6.3.1 Description 33

6.3.2 Pre-conditions 33

6.3.3 Service Flows 34

6.3.4 Post-conditions 34

6.3.5 Existing features partly or fully covering the use case functionality 34

6.3.6 Potential New Requirements needed to support the use case 34

6.3.6.1 Potential KPI Requirements 34

6.4 AI model management as a Service 35

6.4.1 Description 35

6.4.2 Pre-conditions 35

6.4.3 Service Flows 36

6.4.4 Post-conditions 36

6.4.5 Existing features partly or fully covering the use case functionality 36

6.4.6 Potential New Requirements needed to support the use case 36

6.5 AI/ML based Automotive Networked Systems 36

6.5.1 Description 36

6.5.2 Pre-conditions 38

6.5.3 Service Flows 38

6.5.4 Post-conditions 39

6.5.5 Existing features partly or fully covering the use case functionality 40

6.5.6 Potential New Requirements needed to support the use case 40

6.6 Shared AI/ML model monitoring 41

6.6.1 Description 41

6.6.2 Pre-conditions 41

6.6.3 Service Flows 42

6.6.4 Post-conditions 42

6.6.5 Existing features partly or fully covering the use case functionality 42

6.6.6 Potential New Requirements needed to support the use case 42

6.7 Prediction of AI/ML model distribution 42

6.7.1 Description 42

6.7.2 Pre-conditions 43

6.7.3a Service Flows for Prediction based on monitoring function of the 5G network 43

6.7.3b Service Flows for Prediction based on NWDAF analytics function of the 5G network 43

6.7.4 Post-conditions 44

6.7.5 Existing features partly or fully covering the use case functionality 44

6.7.6 Potential Requirements 44

7 Distributed/Federated Learning over 5G system 44

7.1 Uncompressed Federated Learning for image recognition 44

7.1.1 Description 44

7.1.2 Pre-conditions 46

7.1.3 Service Flows 46

7.1.4 Post-conditions 47

7.1.5 Existing features partly or fully covering the use case functionality 47

7.1.6 Potential New Requirements needed to support the use case 47

7.1.6.1 Potential KPI Requirements 47

7.2 Compressed Federated Learning for image/video processing 48

7.2.1 Description 48

7.2.2 Pre-conditions 50

7.2.3 Service Flows 50

7.2.4 Post-conditions 50

7.2.5 Existing features partly or fully covering the use case functionality 50

7.2.6 Potential New Requirements needed to support the use case 52

7.3 Data Transfer Disturbance in Multi-agent multi-device ML Operations 52

7.3.1 Description 52

7.3.2 Pre-conditions 54

7.3.3 Service Flows 55

7.3.4 Post-conditions 56

7.3.5 Existing features partly or fully covering the use case functionality 57

7.3.6 Potential New Requirements needed to support the use case 57

7.4 Group Performance “Flocking” Use Case 59

7.4.1 Description 59

7.4.2 Pre-conditions 60

7.4.3 Service Flows 60

7.4.4 Post-conditions 61

7.4.5 Existing features partly or fully covering the use case functionality 61

7.4.6 Potential New Requirements needed to support the use case 61

8 Consolidated potential requirements 61

8.1 KPI for AMMT services 61

8.2 Functional requirements for AMMT services 63

9 Conclusion and recommendations 64

Annex A: Introduction to AI/ML models 66

A.1 AI and ML 66

A.2 Deep neural network 69

A.3 Training and inference 72

A.4 Widely-used DNN models and algorithms 75

Annex B: General principle of split AI/ML operation between AI/ML endpoints 83

Annex C: General principle of AI/ML model/data distribution and sharing over 5G system 95

Annex D: General principle of Distributed/Federated Learning over 5G system 100

Annex E: Change history 109

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

# Introduction

This document covers use cases and potential requirements for 5G system support of Artificial Intelligence (AI)/Machine Learning (ML) model distribution and transfer (download, upload, updates, etc.). The TR on AI/ML services includes three aspects: AI/ML operation splitting between AI/ML endpoints, AI/ML model/data distribution and sharing over 5G system, distributed/Federated Learning over 5G system.

# 1 Scope

This report captures the study of the use cases and the potential performance requirements for 5G system support of Artificial Intelligence (AI)/Machine Learning (ML) model distribution and transfer (download, upload, updates, etc.), and identifies traffic characteristics of AI/ML model distribution, transfer and training for various applications, e.g. video/speech recognition, robot control, automotive, other verticals.

The aspects addressed include:

* AI/ML operation splitting between AI/ML endpoints;
* AI/ML model/data distribution and sharing over 5G system;
* Distributed/Federated Learning over 5G system.

Study of the AI/ML models themselves are not in the scope of the TR.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TR 22.891, Feasibility Study on New Services and Markets Technology Enablers

[3] 3GPP TR 22.863, Feasibility study on new services and markets technology enablers for enhanced mobile broadband

[4] 3GPP TS 22.261, Service requirements for the 5G system

[5] 3GPP TS 22.104, Service requirements for cyber-physical control applications in vertical domains

[6] 3GPP TS 23.273, 5G System (5GS) Location Services (LCS); Stage 2

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, in Proc. NIPS, 2012, pp. 1097–1105.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556

[9] C. Szegedy, et al., “Going deeper with convolutions”, in Proc. CVPR, 2015, pp. 1-9.

[10] Zhi Zhou, Xu Chen, [En Li](https://ieeexplore.ieee.org/author/37086935882), [Liekang Zeng](https://ieeexplore.ieee.org/author/37086937054), [Ke Luo](https://ieeexplore.ieee.org/author/37086382861), [Junshan Zhang](https://ieeexplore.ieee.org/author/37281464300), “Edge intelligence: Paving the last mile of artificial intelligence with edge computing”, Proceeding of the IEEE, 2019, Volume 107, Issue 8.

[11] [Jiasi Chen, Xukan Ran](https://ieeexplore.ieee.org/author/37596809500), “Deep learning with edge computing: A review”, Proceeding of the IEEE, 2019, Volume 107, Issue 8.

[12] I. Stoica et al., “A Berkeley view of systems challenges for AI”, 2017, arXiv:1712.05855. [Online]. Available: https://arxiv.org/abs/1712.05855

[13] Y. Kang et al., “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge”, ACM SIGPLAN Notices, vol. 52, no. 4, pp. 615–629, 2017.

[14] E. Li, Z. Zhou, and X. Chen, “Edge intelligence: On-demand deep learning model co-inference with device-edge synergy”, in Proc. Workshop Mobile Edge Commun. (MECOMM), 2018, pp. 31–36.

[15] 3GPP TR 38.913, Study on Scenarios and Requirements for Next Generation Access Technologies (Release 15)

[16] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, “A survey of research on cloud robotics and automation,” IEEE Transactions on automation science and engineering, vol. 12, no. 2, pp. 398–409, 2015.

[17] Huaijiang Zhu, Manali Sharma, Kai Pfeiffer, Marco Mezzavilla, Jia Shen, Sundeep Rangan, and Ludovic Righetti, “Enabling Remote Whole-body Control with 5G Edge Computing”, to appear, in Proc. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. Available at: https://arxiv.org/pdf/2008.08243.pdf

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE CVPR, Jun. 2016, pp. 770-778.

[19] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Online]. Available: https://arxiv.org/abs/1704.04861

[20] B. Taylor, V. S.Marco, W. Wolff, Y. Elkhatib, and Z. Wang, “Adaptive deep learning model selection on embedded systems,” in Proc. ACM LCTES, 2018, pp. 31–43.

[21] G. Shu, W. Liu, X. Zheng, and J. Li, “IF-CNN: Image-aware inference framework for CNN with the collaboration of mobile devices and cloud”, IEEE Access, vol. 6, pp. 621–633, 2018.

[22] D. Stamoulis et al., “Designing adaptive neural networks for energy-constrained image classification”, in Proc. ACM ICCAD, 2018, Art. no. 23.

[23] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, In ICML., 2015.

[24] C.-J. Wu et al., “Machine learning at facebook: Understanding inference at the edge,” in Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA), Feb. 2019, pp. 331–344.

[25] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel S. Emer, “Efficient processing of deep neural networks: A tutorial and survey”, Proceeding of the IEEE, 2017, Volume 105, Issue 12.

[26] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[27] “[An All-Neural On-Device Speech Recognizer](http://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html)”, March 12, 2019, Posted by Johan Schalkwyk, https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

[28] [Yanzhang He](https://ieeexplore.ieee.org/author/38666436000), etc., “[Streaming End-to-end Speech Recognition for Mobile Devices](https://ieeexplore.ieee.org/document/8682336/)”, [2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)](https://ieeexplore.ieee.org/xpl/conhome/8671773/proceeding)

[29] 3GPP TS 22.243: "Speech recognition framework for automated voice services; Stage 1".

[30] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data”, Proc. of the International Confe rence on Artificial Intelligence and Statistics, Apr. 20 17. [Online]. Available: <https://arxiv.org/abs/1602.05629>

[31] “Federated Learning”, https://justmachinelearning.com/2019/03/10/federated-learning/

[32] T. Nishio and R. Yonetani, “Client selection for federated learning with heterogeneous resources in mobile edge”, 2018, arXiv:1804.08333. [Online]. Available: https://arxiv.org/abs/1804.08333

[33] E. Park et al., “Big/little deep neural network for ultra low power inference”, in Proc. 10th Int. Conf. Hardw./Softw. Codesign Syst. Synth., 2015, pp. 124–132.

[34] [Nguyen H. Tran](https://ieeexplore.ieee.org/author/38182024500); [Wei Bao](https://ieeexplore.ieee.org/author/37071870400); [Albert Zomaya](https://ieeexplore.ieee.org/author/37268553900); [Minh N. H. Nguyen](https://ieeexplore.ieee.org/author/37085814808); [Choong Seon Hong](https://ieeexplore.ieee.org/author/37287529400), “[Federated Learning over Wireless Networks: Optimization Model Design and Analysis](https://ieeexplore.ieee.org/document/8737464/)”, In proc. [IEEE INFOCOM 2019 - IEEE Conference on Computer Communications](https://ieeexplore.ieee.org/xpl/conhome/8732929/proceeding)

[35] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. “EIE: efficient inference engine on compressed deep neural network”, In 43rd International Symposium on Computer Architecture, IEEE Press, 243–254.

[36] V. Sze, “Efficient Computing for Deep Learning, AI and Robotics,” Dept EECS, MIT, Available online at <https://lexfridman.com/files/slides/2020_01_15_vivienne_sze_efficient_computing.pdf>

[37] V. Sze, Y. Chen, “Efficient Processing of Deep Neural Networks: A Tutorial and Survey” Proc. of IEEE, 2017, Available online at: <https://www.semanticscholar.org/paper/Efficient-Processing-of-Deep-Neural-Networks%3A-A-and-Sze-Chen/3f116042f50a499ab794bcc1255915bee507413c>

[38] Stanford University, CS231n – Lecture 5-7: CNN, Training NNs, Available at YouTube.com

[39] S. Han, J. Pool, J. Tran, and W, J. Dally, "Learning both weights and connections for efficient neural networks", NIPS, May 2015

[40] P. A. Merolla, et al., “A million spikingneuron integrated circuit with a scalable communication network and interface”,Science, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[41] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” J. Mach. Learn. Res., vol. 12 pp. 2493–2537, Aug. 2011.

[42] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep convolutionalneural networks for LVCSR”, in Proc. ICASSP, 2013, pp. 8614–8618.

[43] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey”, J. Artif. Intell. Res., vol. 4, no. 1, pp. 237–285, Jan. 1996.

[44] 3 AI Trends for Enterprise Computing. [Online]. Available: https://www.gartner.com/smarterwithgartner/3-ai-trends-for-enterprise-computing/

[45] [Shiming Ge](https://ieeexplore.ieee.org/author/37086013318); [Zhao Luo](https://ieeexplore.ieee.org/author/37086057228); [Shengwei Zhao](https://ieeexplore.ieee.org/author/37086052498); [Xin Jin](https://ieeexplore.ieee.org/author/37086047196); [Xiao-Yu Zhang](https://ieeexplore.ieee.org/author/37086056313), “[Compressing deep neural networks for efficient visual inference](https://ieeexplore.ieee.org/document/8019465/)”, In proc. [2017 IEEE International Conference on Multimedia and Expo (ICME)](https://ieeexplore.ieee.org/xpl/conhome/8014303/proceeding)

[46] 3GPP TS 22.186, Enhancement of 3GPP support for V2X scenarios; Stage 1 (Release 16) v16.2.0

[47] “Develop Smaller Speech Recognition Models with NVIDIA’s NeMo Framework”, <https://developer.nvidia.com/blog/develop-smaller-speech-recognition-models-with-nvidias-nemo-framework/>

[48] 3GPP TS 23.501, System architecture for the 5G System (5GS)

[49] 3GPP TS 23.502, Procedures for the 5G System (5GS)

[50] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”

[51] J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement”

[52] W. Sun, Z. Chen, “Learned Image Downscaling for Upscaling using Content Adaptive Resampler”

[53] C. Ledig et al., “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”

[54] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Available online: https://arxiv.org/abs/1704.04861]

[55] SSD-ResNet34 - https://github.com/IntelAI/models/tree/master/benchmarks/object\_detection/tensorflow/ssd-resnet34

[56] MLCommons Mobile Inference Benchmark v0.7 - https://mlcommons.org/en/inference-mobile-07/

[57] MASK C-RNN - https://arxiv.org/abs/1703.06870

[58] <https://ai.facebook.com/blog/dlrm-an-advanced-open-source-deep-learning-recommendation-model/>

[59] 3GPP TR 28.809, Study on enhancement of management data analytics

[60] Mingzhe Chen, "A Joint Learning and Communications Framework for Federated Learning over Wireless Networks", Oct 2020

# 3 Definitions, symbols and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

**communication service availability**: percentage value of the amount of time the end-to-end communication service is delivered according to an agreed QoS, divided by the amount of time the system is expected to deliver the end-to-end service according to the specification in a specific area.

NOTE 1: This definition was taken from TS 22.261 [4].

**End-to-End Latency:** the time that takes to transfer a given piece of information from a source to a destination, measured at the communication interface, from the moment it is transmitted by the source to the moment it is successfully received at the destination.

NOTE 2: The end point in "end-to-end" is assumed to be the communication service interface.

NOTE 3: This definition was taken from TS 22.261 [4].

**reliability**: in the context of network layer packet transmissions, percentage value of the amount of sent network layer packets successfully delivered to a given system entity within the time constraint required by the targeted service, divided by the total number of sent network layer packets.

NOTE 4: This definition was taken from TS 22.261 [4].

**user experienced data rate:** the minimum data rate required to achieve a sufficient quality experience, with the exception of scenario for broadcast like services where the given value is the maximum that is needed.

NOTE 5: This definition was taken from TS 22.261 [4].

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

AI Artificial Intelligence

CNN Convolution Neural Network

DNN Deep Neural Network

FL Federated Learning

GPU Graphics Processing Units

IDC Internet Data Center

ML Machine Learning

# 4 Overview

Artificial Intelligence (AI)/Machine Learning (ML) is being used in a range of application domains across industry sectors. In mobile communications systems, mobile devices (e.g. smartphones, automotive, robots) are increasingly replacing conventional algorithms (e.g. speech recognition, image recognition, video processing) with AI/ML models to enable applications. The 5G system can at least support three types of AI/ML operations:

* AI/ML operation splitting between AI/ML endpoints;
* AI/ML model/data distribution and sharing over 5G system;
* Distributed/Federated Learning over 5G system.

The scheme of split AI/ML inference can be depicted as in Figure 4-1. The AI/ML operation/model is split into multiple parts according to the current task and environment. The intention is to offload the computation-intensive, energy-intensive parts to network endpoints, whereas leave the privacy-sensitive and delay-sensitive parts at the end device. The device executes the operation/model up to a specific part/layer and then sends the intermediate data to the network endpoint. The network endpoint executes the remaining parts/layers and feeds the inference results back to the device.

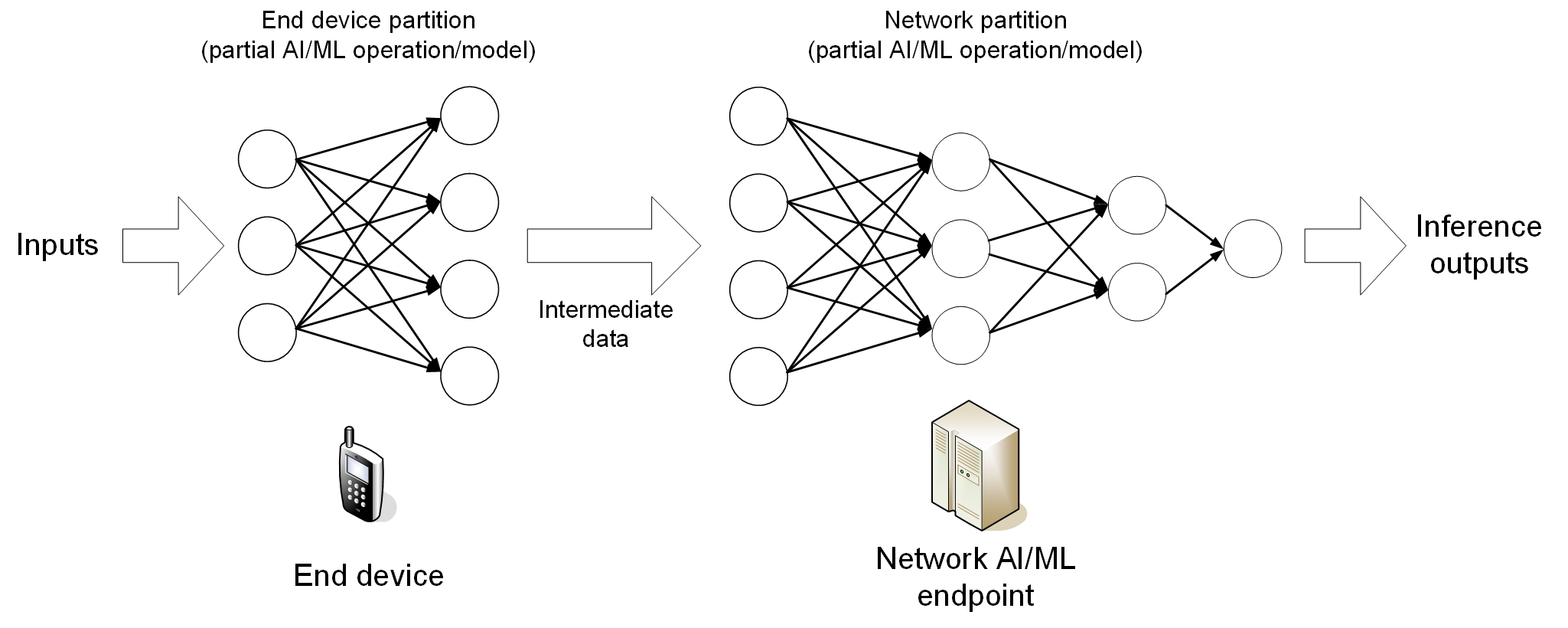


Figure 4-1. Example of split AI/ML inference

The scheme of AI/ML model distribution can be depicted as in Figure 4-2. Multi-functional mobile terminals might need to switch the AI/ML model in response to task and environment variations. The condition of adaptive model selection is that the models to be selected are available for the mobile device. However, given the fact that the AI/ML models are becoming increasingly diverse, and with the limited storage resource in a UE, it can be determined to not pre-load all candidate AI/ML models on-board. Online model distribution (i.e. new model downloading) is needed, in which an AI/ML model can be distributed from a NW endpoint to the devices when they need it to adapt to the changed AI/ML tasks and environments. For this purpose, the model performance at the UE needs to be monitored constantly.

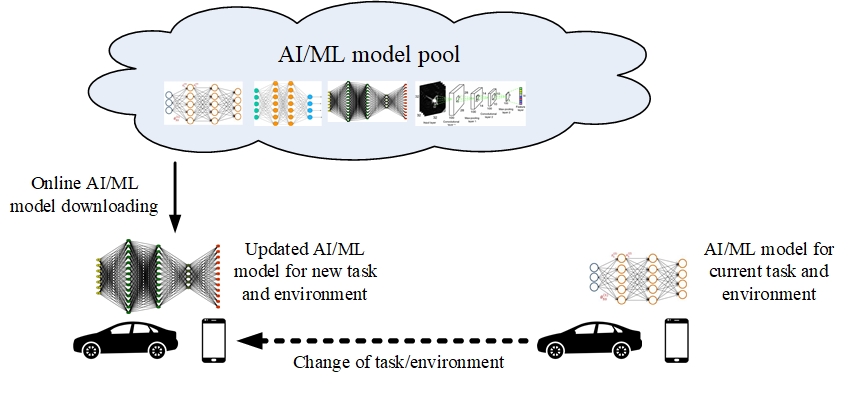


Figure 4-2. AI/ML model downloading over 5G system

The scheme of Federated Learning (FL) can be depicted as in Figure 4-3. The cloud server trains a global model by aggregating local models partially-trained by each end devices. Within each training iteration, a UE performs the training based on the model downloaded from the AI server using the local training data. Then the UE reports the interim training results to the cloud server via 5G UL channels. The server aggregates the interim training results from the UEs and updates the global model. The updated global model is then distributed back to the UEs and the UEs can perform the training for the next iteration.

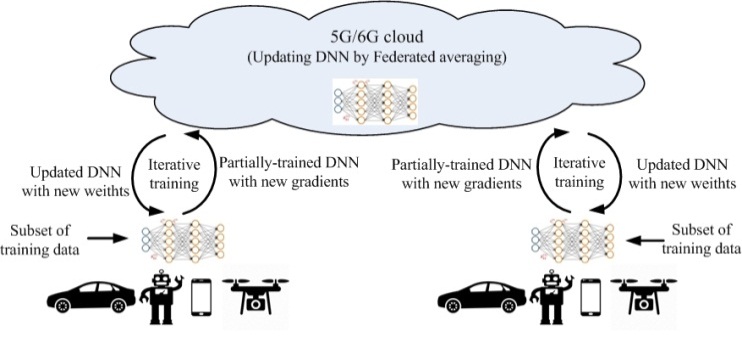


Figure 4-3. Federated Learning over 5G system

# 5 Split AI/ML operation between AI/ML endpoints

## 5.1 Split AI/ML image recognition

### 5.1.1 Description

The AI/ML-based mobile applications are increasingly computation-intensive, memory-consuming and power-consuming. Meanwhile end devices usually have stringent energy consumption, compute and memory limitations for running a complete offline AI/ML inference on-board. Many AI/ML applications, e.g. image recognition, currently intent to offload the inference processing from mobile devices to internet datacenters (IDC). For example, photos shot by a smartphone are often processed in a cloud AI/ML server before shown to the user who shot them. However, the cloud-based AI/ML inference tasks need to take into account the computation pressure at IDCs, required data rate/latency and privacy protection requirement.

Image and video are the biggest data on today’s Internet. Videos account for over 70% of daily Internet traffic [4]. Convolutional Neural Network (CNN) models have be widely used for image/video recognition tasks on mobile devices, e.g. image classification, image segmentation, object localization and detection, face authentication, action recognition, enhanced photography, VR/AR, video games. Meanwhile, CNN model inference requires an intensive computation and storage resource. For example, AlexNet [7], VGG-16 [8] and GoogleNet [9] require 724M, 15.5G and 1.43G MACs (multiply-add computation) respectively for a typical image classification task.

Many references [10-14] have shown that AI/ML inference for image processing with device-network synergy can alleviate the pressure of computation, memory footprint, storage, power and required data rate on devices, reduce end-to-end latency and energy consumption, and improve the end-to-end accuracy, efficiency and privacy when compared to the local execution approach on either side. The scheme of split AI/ML image recognition can be depicted in Figure 5.1.1-1. The CNN is split into two parts according to the current image recognition task and environment. The intention is to offload the computation-intensive, energy-intensive parts to network server, whereas leave the privacy-sensitive and delay- sensitive parts at the end device. The device executes the inference up to a specific CNN layer and sends the intermediate data to the network server. The network server runs through the remaining CNN layers. While the model is developed or invocated, the split AI/ML operation is based on the legacy model.

Due to the characteristics of some algorithms in the model training phase, a model has a certain degree of robustness[xx-xy]. Therefore, if there are errors in the intermediate data transmission, the model has a certain tolerance and can still guarantee the accuracy of the inference results. Since the inference result needs to be forwarded to the UE, the reliability of the inference result transmission needs to be guaranteed.



Figure 5.1.1-1. Example of split AI/ML image recognition

The split AI/ML image recognition algorithms can be analyzed based on the computation and data characteristics of the layers in the CNN. As shown in figure 5.1.1-2 and 5.1.1-3 (based on figures adopted from [13]), the intermediate data size transferred from one CNN layer to the next depends on the location of the split point. Hence, the required UL data rate is related to the model split point and the frame rate for the image recognition, as also observed by [13-14]. For example, assuming images from a video stream with 30 frames per second (FPS) need to be classified, the required UL data rate for different split points ranges from 4.8 to 65 Mbit/s (listed in Table 5.1.1-1). The result is based on the 227×227 input images. In case of images with a higher resolution, higher data rates would be required.

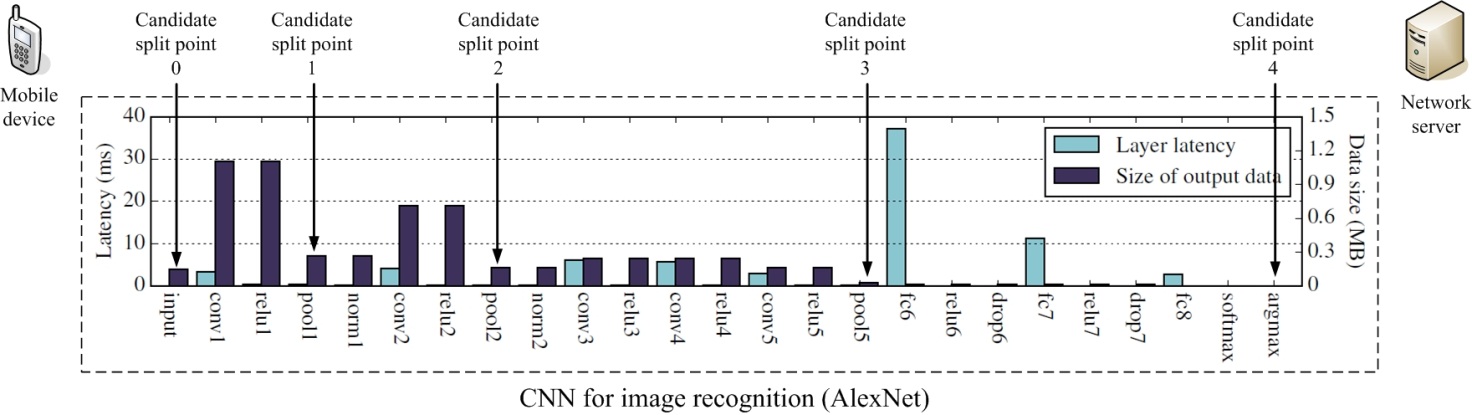


Figure 5.1.1-2. Layer-level computation/communication resource evaluation for an AlexNet model

Table 5.1.1-1: Required UL data rate for different split points of AlexNet model for video recognition @30FPS

|  |  |  |
| --- | --- | --- |
| **Split point** | **Approximate output data size (MByte)** | **Required UL data rate (Mbit/s)** |
| Candidate split point 0  (Cloud-based inference) | 0.15 | 36 |
| Candidate split point 1  (after pool1 layer) | 0.27 | 65 |
| Candidate split point 2  (after pool2 layer) | 0.17 | 41 |
| Candidate split point 3  (after pool5 layer) | 0.02 | 4.8 |
| Candidate split point 4  (Device-based inference) | N/A | N/A |

VGG-16 is another widely-used CNN model for image recognition. Still assuming images from a video stream with 30 FPS need to be classified, the required UL data rate for different split points ranges from 24 to 720 Mbit/s (listed in Table 5.1.1-2).

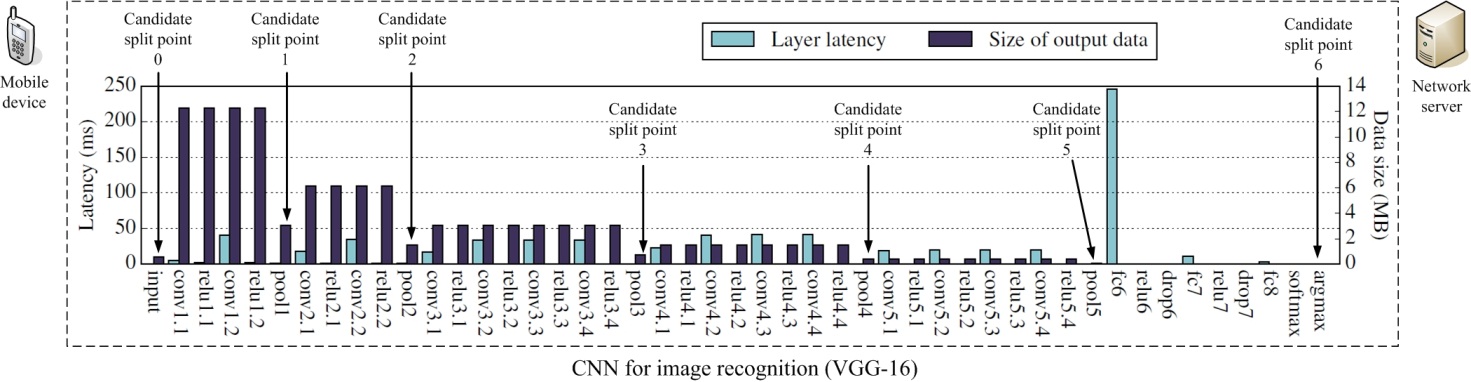


Figure 5.1.1-3. Layer-level computation/communication resource evaluation for a VGG-16 model

Table 5.1.1-2: Required user experienced UL data rate for different split points of VGG-16 model @30FPS

|  |  |  |
| --- | --- | --- |
| **Split point** | **Approximate output data size (MByte)** | **Required user experienced UL data rate (Mbit/s)** |
| Candidate split point 0  (Cloud-based inference) | 0.6 | 145 |
| Candidate split point 1  (after pool1 layer) | 3 | 720 |
| Candidate split point 2  (after pool2 layer) | 1.5 | 360 |
| Candidate split point 3  (after pool3 layer) | 0.8 | 192 |
| Candidate split point 4  (after pool4 layer) | 0.5 | 120 |
| Candidate split point 5  (after pool5 layer) | 0.1 | 24 |
| Candidate split point 6  (Device-based inference) | N/A | N/A |

### 5.1.2 Pre-conditions

The involved AI/ML endpoints (e.g. UE, AI/ML cloud/edge server) run applications providing the capability of AI/ML model inference for image recognition, and support the split AI/ML image recognition operation.

The 5G system has the ability to provide 5G network related information to the AI/ML server.

### 5.1.3 Service Flows

1. The AI/ML based image recognition application is requested by the user to start recognizing the image/video shot by the UE.
2. Under the determined split mode and split point, the AI/ML based image recognition application in an involved AI/ML endpoint executes the allocated part of AI/ML model, and sends the intermediate data to the next endpoint in the AI/ML pipeline.
3. After all the involved AI/ML endpoints finish the co-inference, the image recognition results are fed to the user using the results.
4. The AI/ML based image recognition applications in the endpoints perform the split image recognition until the image recognition task is terminated.

Redo Step 3) and 4) for split mode/point re-selection/switching if needed to adapt to the changing conditions.

### 5.1.4 Post-conditions

The objects in the input images or videos are recognized and the recognition accuracy and latency need to be guaranteed.

The image recognition task can be completed under the available computation and energy resource of the UE. And the consumed the computation, communication and energy resources over the AI/ML endpoints are optimized.

### 5.1.5 Existing features partly or fully covering the use case functionality

This use case mainly requires high data rate together with low latency. The high data rate requirements to 5G system are listed in Clause 7.1 and 7.6 of TS22.261 [4]. As in Table 7.1-1 of [4], 300Mbps DL experienced data rate and 50Mbps UL experienced data rate are required in dense urban scenario, and 1Gbps DL experienced data rate and 500Mbps UL experienced data rate are required in indoor hotspot scenario. As in Table 7.6.1-1 of [4], cloud/edge/split rendering- related data transmission requires up to 0.1Gbps data rate with [5-10]ms latency countrywide.

### 5.1.6 Potential New Requirements needed to support the use case

The “image recognition latency” can be defined as the latency from the image is captured to the recognition results of the image are output to the user application, which was not specially addressed in [4][15]. Following the principle of analyzing the latency and data rate requirements of split image recognition introduced in Section 5.1.1, the image recognition latency is related to the user application the recognition is used for.

Computer vision and image recognition have been widely used for many important mobile applications such as object recognition, photo enhancements, intelligent video surveillance, mobile AR, remote-controlled automotive, industrial control and robotics. The image recognition is usually a step of the processing pipeline of the application. And the recognition latency is a part of the end-to-end latency, as depicted in Figure 5.1.6-1.

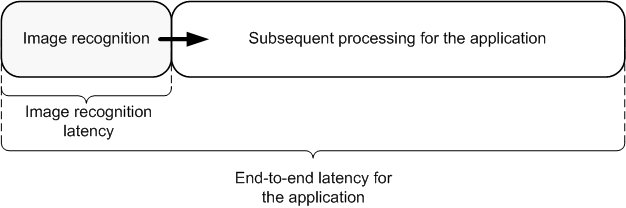


Figure 5.1.6-1. Image recognition latency is a part of end-to-end latency

For example, if the image recognition results are just used for the object recognition e.g. unknown object recognition for smartphone user or criminal searching in database for intelligent security, it is acceptable that the image recognition is finished in seconds. If the image recognition result is used as an input to another time-sensitive application, e.g. AR display/gaming, remote-controlled automotive, industrial control and robotics, a much more stringent latency will be required. Based on the end-to-end latency requirements of the applications, the image recognition latency requirement can be derived, as listed in Table 5.1.6-1.

#### 5.1.6.1 Potential KPI Requirements

The potential KPI requirements needed to support the use case include:

[P.R.5.1-001] The 5G system shall support intermediate data uploading for split image recognition with a maximum latency as given in Table 5.1.6.1-1.

[P.R.5.1-002] The 5G system shall support intermediate data uploading for split image recognition with a user experienced UL data rate as given in Table 5.1.6.1-1.

[P.R.5.1-003] The 5G system shall support intermediate data uploading for split image recognition with communication service availability not lower than 99.999 %.

[P.R.5.1-004] The 5G system shall support inference results downloading for split image recognition with reliability not lower than 99.999 % and intermediate data uploading for split image recognition with reliability not lower than 99.9%.

NOTE: Above requirements apply to intermediate data with a maximum size of 0.27MB (for AlexNet model) or 1.5 MB (for VGG-16 model).

Table 5.1.6.1-1: Image recognition latency and UL data rate for intermediate data uploading

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User application** | **Latency: maximum** | | | **User experienced UL data rate** | |
| **End-to-end latency** | **Image recognition latency** | **Intermediate data uploading latency** | **AlexNet**  **(Fig. 5.1.1-1, see note 4)** | **VGG-16**  **(Fig. 5.1.1-2, see note 4)** |
| One-shot object recognition at smartphone | Several seconds | ~1s | ~100ms | 1.6~21.6Mbit/s | 8~240Mbit/s |
| Person identification in security surveillance system | Several seconds | ~1s | ~100ms | 1.6~21.6Mbit/s | 8~240Mbit/s |
| Photo enhancements at smartphone | Several seconds | ~1s | ~100ms | 1.6~21.6Mbit/s | 8~240Mbit/s |
| Video recognition | Several seconds | 33ms@30FPS | ~10ms | 16~216Mbit/s | 80Mbit/s~2.4Gbit/s |
| AR display/gaming | 7~15ms (see note 1) | <5ms | 2ms | 80Mbit/s~1.08Gbit/s | 0.4~12Gbit/s |
| Remote driving | 10ms (see note 2) | <5ms | 2ms | 80Mbit/s~1.08Gbit/s | 0.4~12Gbit/s |
| Remote-controlled robotics | 10~100ms (see note 3) | <5ms | 2ms | 80Mbit/s~1.08Gbit/s | 0.4~12Gbit/s |
| NOTE 1: According to [4][5], the VR motion-to-photon latency is in the range of 5-15ms.  NOTE 2: According to [46] the one-way latency required for remote driving is 5ms. The round-trip latency is assumed to be 10ms.  NOTE 3: According to [5] the end-to-end latency required for video-operated remote-controlled robotics is 10~100ms.  NOTE 4: As listed in table 5.1.1-1 and table 5.1.1-2, the intermediate data size for the split points of AlexNet and VGG-16 is 0.02 ~ 0.27MByte and 0.1 ~ 3MByte respectively. | | | | | |

## 5.2 Enhanced media recognition: Deep Learning Based Vision Applications

### 5.2.1 Description

A tourist is wandering around a city and discovering the attractions and sights of the city. The user sees a beautiful object and she decides to shoot a video of the object. The application uses deep learning algorithms to process the video and identify the object of interest and provide historical information about it to the user. Furthermore, the application uses deep learning to reconstruct a 3D model of the object of interest by using the captured 2D video.

As an example, we investigate Feature Pyramid Network (FPN)-based object detection approaches. These networks are usually composed of a backbone FPN and a head that performs task-specific inference. The FPN processes the input images at different scales to allow for the detection of small-scale and large-scale features. The head may for instance segment the objects, infer a bounding box for the objects, or classify the objects.

The FPN backbone constitutes the most complex portion of the network and lends itself to be offloaded to the edge/cloud. The backbone is a common part to a wide range of networks that can perform different tasks. The produced feature maps can then be sent back to the UE for task-specific inference.

A breakdown of the network architecture is shown in the following figure:

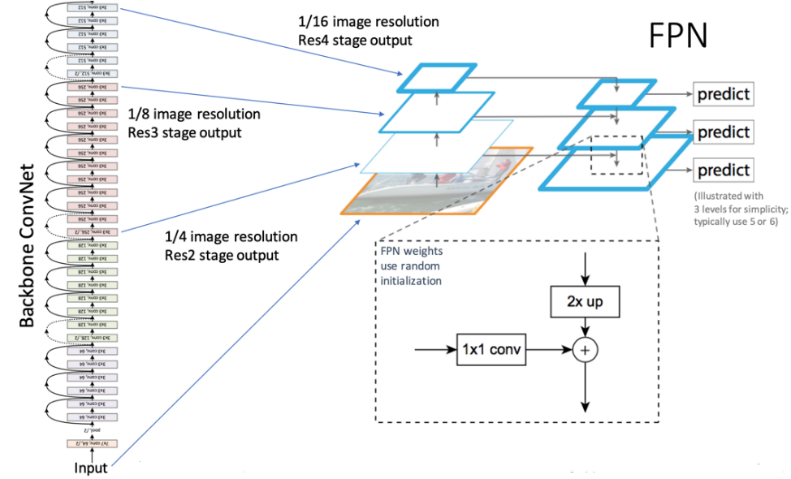


Figure 5.2.1-1. Example Multi-Task Network

As shown in figure 5.2.1-1, a classical CNN architecture is used as the core. The FPN is used to extract features at different scales of the image, making it scale-invariant. The prediction tasks constitute the head of the network. By plugging in different network heads, different AI tasks can be performed. This makes the network a Multi-Task Network (MTN). For example, a Region Proposal Network can be appended to detect and frame objects in the input sequence by outputting bounding boxes. Other Task-specific Heads can be appended to detect humans and poses, classify objects, track objects, etc.

### 5.2.2 Pre-conditions

The user wants to receive instantaneous information and reconstruction to enhance their experience. The user’s device is battery operated.

### 5.2.3 Service Flows

1. User opens their camera app and starts shooting a video
2. Application pre-processes the video to prepare it for inference

NOTE: Pre-processing is assumed to be not very intensive in this use case, e.g. downscaling the video, adjusting the frame rate, extracting simple features.

1. The application streams the extracted features and/or the video to the edge/cloud for processing.
2. The network performs the split-inference (e.g. only running the backbone) and streams the results back to the client
3. The application runs task-specific inference to solve the specific task of interest (e.g. object detection, tracking, …)
4. The application uses the inferred labels and classes to enhance the user’s view

### 5.2.4 Post-conditions

The user gets enhanced information extracted from the video about the object of interest that the user was shooting a video of.

### 5.2.5 Existing features partly or fully covering the use case functionality

None.

### 5.2.6 Potential New Requirements needed to support the use case

The potential KPI requirements to support the use case are as given in Table 5.2.6.1-1, based on two examples of object detection models/algorithms:

- uplink streaming latency not higher than [100-200ms] and a user experienced UL data rate of [100-1500] kbit/s;

- downlink streaming latency not higher than [100-500ms] and a user experienced DL data rate of [32-150] Mbit/s;

Table 5.2.6.1-1: Recognition: latency breakdown and user experienced UL/DL data rates

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Recognition Task** | **Latency: maximum**  **(see note 4)** | **User experienced DL data rate** | | **User experienced UL data rate** | |
| **Faster R-CNN [50]**  **(see note 1)** | **YOLOv3 [51]**  **(see note 2)** | **Faster R-CNN** | **YOLOv3** |
| Uplink Streaming | 100-200ms |  |  | 100-1000 kbit/s | 200-1500 kbit/s |
| Generic FPN Inference | 100-500ms | FPN:  4-10fps Sum(Pi)~1MB/frame 32-100Mbit/s uncompressed  (see note 3)  Compression factor 10~100 | Multiple scale (similar to FPN):  1.5 MB feature map/frame  40-150 Mbit/s uncompressed  Compression factor 10~100 |  |  |
| Object Classification | 20-50ms | Performed on UE | Performed on UE |  |  |
| Bounding Box Detection | 20-50ms | Performed on UE | Performed on UE |  |  |
| Object Tracking | 50-150ms | Performed on UE | Performed on UE |  |  |
| Enhanced Information Retrieval |  | Few kBytes per request | | Few kBytes per request | |
| Overlay Rendering | 10ms | Performed on UE | | Performed on UE | |
| Note 1: Faster R-CNN uses an input image size of 3x224x224. The video is downscaled on the UE to that target resolution and then compressed (e.g. using HEVC) and streamed to the edge for further processing.  Note 2: YOLOv3 uses an input image size of 3x416x416. The captured video is downscaled on the UE to the target resolution and compressed prior to streaming to the edge.  Note 3: Faster R-CNN uses an FPN with ResNet 101 as backbone; thus resulting in feature maps {P2=(256x56x56), P3=(256x28x2), P4=(256x14x14), P5=(256x7x7)}.  Note 4: the latency estimates assume an overall latency of around 1s from a user pointing at an object until overlay information is displayed to the user. | | | | | |

## 5.3 Media quality enhancement: Video streaming upgrade

### 5.3.1 Description

A user is playing a VR game on the cloud using their VR headset. The game is being rendered on the cloud and streamed down to the UE. The user wants to enjoy an immersive gaming experience which requires very high quality video, e.g. 8K per eye at 120 frames per second. The cloud game server can only produce 4K video data due to hardware, load, and networking restrictions. AI is used to upscale the 4K content into 16K content for a better user experience.

The following figure shows an example of such a network:

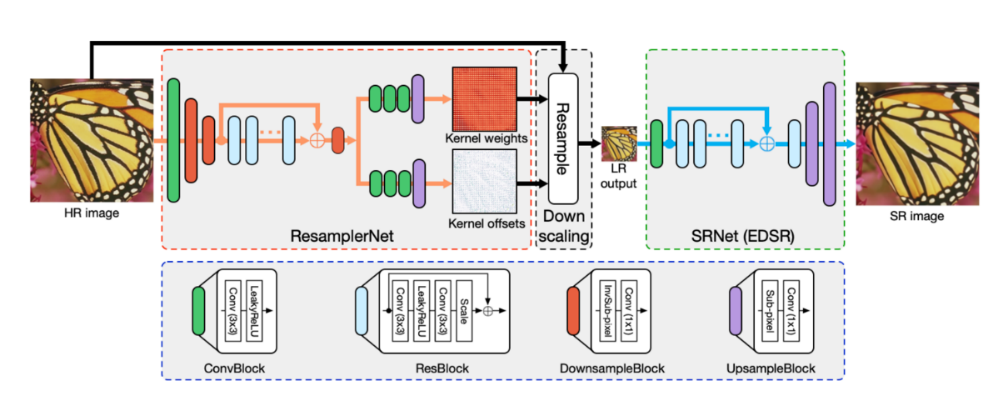


Figure 5.3.1-1. Example DNN-based Down/Up-scaler

The Low Resolution video is streamed to the UE, which will process the video to infer the high resolution version. The down-sampling and up-sampling parts of the network are matched to produce best results. Any updates to the down-sampling part of network would require updates on the UE-side to the up-sampling part of the network. In addition to the LR version of the video, model weights and topology updates may need to be sent to the UE.

The pre-trained models are optimized to the type of content (e.g. sport, cartoons, movie, …) and to the scale factor. As the user switches to a different piece of content, the device will check if the corresponding pre-trained model weights are already downloaded in the cache. If not available, the UE will download the pre-trained model weights for the selected piece of content and based on the desired up-scaling factor. The new content is shown to the user in less than 3 seconds from the user selecting it.

### 5.3.2 Pre-conditions

The remote gaming server generates streams metadata together with the stream that is extracted by running AI autoencoder on the originally captured content.

### 5.3.3 Service Flows

1. Users starts a cloud VR gaming session on their HMD
2. The game is launched on the cloud server and the game can start
3. The cloud server renders and captures the content and downscales it to 4K video
4. The cloud server also runs a DNN to produce a metadata stream that will be used for upscaling
5. The UE uses a reverse DNN network to upscale the received 4K stream to 16K. The input to the DNN is the 4K video and the metadata stream.
6. The UE renders the high quality 16K view on the HMD.

### 5.3.4 Post-conditions

The user enjoys a high-quality VR experience.

### 5.3.5 Existing features partly or fully covering the use case functionality

None.

### 5.3.6 Potential New Requirements needed to support the use case

The potential KPI requirements to support the use case are:

[P.R.5.3-001] The 5G system shall support the delivery of pre-trained model weights and weight updates with a latency no more than 3 seconds.

NOTE 1: the model download time takes into account acceptable channel switch time, which generally should not exceed 2-3 seconds.

The following table provides an estimate of the model weight sizes:

Table 5.3.6.1-1: Model Weight Sizes and Data rates

|  |  |  |
| --- | --- | --- |
| **Model Name (see note 2)** | **Model Weight Size (see note 4)** | **Data Rate (see note 3)** |
| CAR 4x [52] | Downsampling: ~40MB  Upsampling: ~170MB | Upsampling model download:  450Mbit/s (downloaded over a period of 3 seconds) |
| SR-GAN 4x [53] | Downsampling: N/A (bicubic)  Upsampling: ~6MB | Upsampling model download:  12Mbit/s (downloaded over a period 3 seconds) |
| SRResNet 4x [53] | Downsampling: N/A (bicubic)  Upsampling: ~6MB | Upsampling model download:  12Mbit/s (downloaded over a period of 3 seconds) |
| NOTE 2: the size of the weights of the pre-trained models depends on the nature of the content and the scaling factor of the video.  NOTE 3: model weight download is triggered by change of content by the user and thus results in bursty traffic by nature with relatively long off periods.  NOTE 4: all model weight sizes are from pre-trained PyTorch models. | | |

## 5.4 Split control for robotics

### 5.4.1 Description

Mobile robots have been playing an increasingly important role in some scenarios, e.g. warehouse, disaster rescue and smart factories [5], thanks to their high mobility. The mobile robots need to work in an ever-changing environment, hence need to perform fast and reliable sensing, planning and controlling. If the corresponding computation is performed on board in the robot, it will require intensive computations which lead to increased requirements of computation capabilities and power consumption. However, a light-weight form factor is always a requirement to the mobile robots working in the real-world environment, which prevents the robots to be equipped with a large number of CPU/GPU units and large-capacity batteries. As the example provided in [17], an advanced commercially available quadruped robot, carries 3kg of batteries of about 650Wh energy, while the high-end GPU consumes more than 250W of power, significantly impacting battery life if such computational power was embedded on the robot.

Offloading computations from robots to the cloud has been studied in many references [16]. Meanwhile relying on either data or code from a network to support the robot’s operation, the designers of autonomous mobile robots have to consider the scenarios where the robots must include capacity of local processing for low-latency responses during periods when network access quality is varying worse.

The resulting system is different from the fully remote-controlled robot system described in [5], in which the planning and controlling are carried out by cloud computing, and the robots only reports the sensing data (incl. video), and receives the control commands. Since the complete cloud computing can hardly meet the latency requirement of the ms-level feedback control loop of some types of mobile robots, e.g. legged robots, the split control of mobile robots is an agreeable solution in this case.

[17] introduces a robot whole-body balance control split over 5G network. The AI inference for the controlling can be split between the robot and the cloud server: As shown in Figure 5.4.1-1, the part which is complex but less susceptible to delays is offloaded to the remote computation in the cloud or edge control server. The low-complexity part which contains the error feedback terms and is latency-critical can be efficiently done by the local computation in the robot. If the robot fails to receive the optimal control from “remote control part” from the cloud/edge control server due to communication delays or packet loss, it can approximate the “remote control part” using pre-computed feedback matrices received previously. And in certain duration, the approximation will still enable the robot to perform feedback control for the tasks approximately and ensure that the robot can still operate.

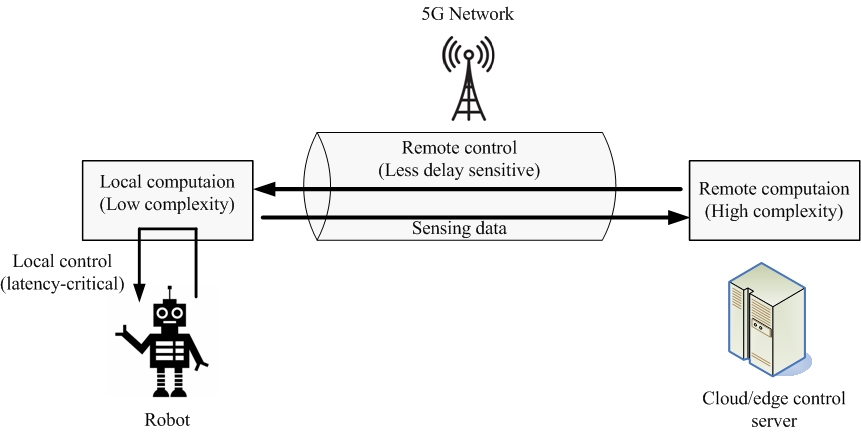
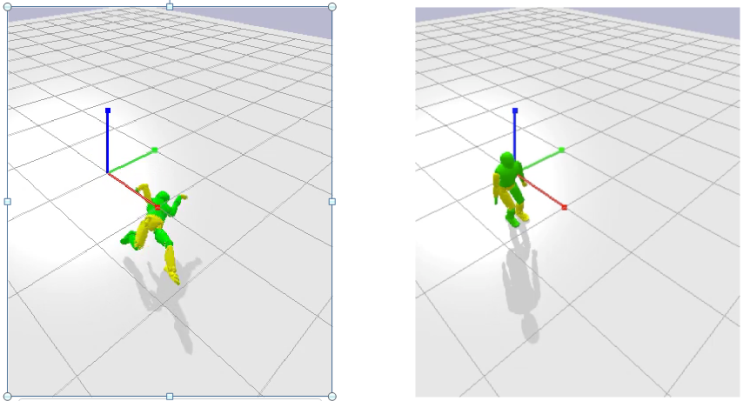


Figure 5.4.1-1. Split control of legged robot over 5G network

The results in [17] show that, in case the robot is completely controlled by a cloud server, the robot cannot finish the walking task if the round-trip latency is larger than 3ms (from sending sensing data to receiving control commands, including processing at cloud/edge). Due to delayed control commands, the robot would fall down (as shown in Figure 5.4.1-2 (a)). However, if the split control is employed, a worse-case 25ms round-trip latency can be sustained, and the robot can still perform the walking task (as shown in Figure 5.4.1-2 (b)).



1. 5G remote control without local control (b) 5G remote control with local control at robot

Figure 5.4.1-2. Simulated performance of robot whole-body balance control over 5G network with 25ms round-trip latency

### 5.4.2 Pre-conditions

The involved AI/ML endpoints (e.g. UE (robot), AI/ML cloud/edge server) run applications providing the capability of AI/ML model inference for robot control task, and support the split robot control operation.

The 5G system has the ability to provide 5G network related information to the AI/ML server.

### 5.4.3 Service Flows

1. The UE (robot) is connected with the cloud/edge control server via 5G network.
2. The split of the control operations on robot side and cloud/edge control server side for the robot control task is determined by the robot or the network.
3. Under the determined split mode and split point, the robot performs the local control computation based on the collected sensing data, and sends the sensing data to the cloud/edge control server. The cloud/edge control server performs the remote control computation, and feeds the outputs back to the robot.
4. The robot controls its motion jointly based on the outputs of local and remote control computations.
5. Start with Step 3) with more control operations, until the robot control task is terminated.

### 5.4.4 Post-conditions

The robot receives the control from local and remote with required accuracy and latency, so to finish the moving tasks, e.g. balance task and walking task.

The robot control task can be completed under the available computation and energy resource of the robot. And the consumed the computation, communication and energy resources over the AI/ML endpoints are optimized.

### 5.4.5 Existing features partly or fully covering the use case functionality

This use case mainly requires high data rate together with low latency. The high data rate requirements to 5G system are listed in Clause 7.1 and 7.6 of TS22.261 [4]. As in Table 7.1-1 of [4], 300Mbps DL experienced data rate and 50Mbps UL experienced data rate are required in dense urban scenario, and 1Gbps DL experienced data rate and 500Mbps UL experienced data rate are required in indoor hotspot scenario. As in Table 7.6.1-1 of [4], cloud/edge/Split rendering-related data transmission requires up to 0.1Gbps data rate with [5-10]ms latency countrywide.

### 5.4.6 Potential New Requirements needed to support the use case

If everything is done on the edge part, the robot needs to send 592B sensing data per control cycle (every millisecond) and receive 200B per control cycle from the “remote controller”, which leads to a UL data rate of 4.7Mbit/s and a DL data rate of 1.6Mbit/s.  However maximum round-trip latency is limited to 3ms, and the one-way latency for downloading “remote control part” needs to be limited to 1ms. If the splitting strategy is followed, downloading the “remote control part” from the cloud/edge control server requires downloading a 40kB data burst per control cycle (every millisecond) as more information is needed to ensure the local controller can take over in case of unexpected latencies, which leads to a user experienced DL data rate of 320Mbit/s. In that case, a maximum round-trip latency of 25ms can be tolerated between each control cycle, and the one-way latency for downloading “remote control part” can be relaxed to 12ms.

This implies a trade-off between DL data rate and latency: Compared with the full control at edge, the split control mode requires a higher user experienced DL data rate, but relaxes the stringent latency requirement. Different from the traditional URLLC services requiring continuous coverage of 5G network which can only be provided with FR1 spectrum, for a 5G operator with FR2 spectrum, the split control for robotics can be offloaded to the 5G mmWave network with non-continuous coverage.

#### 5.4.6.1 Potential KPI Requirements

The potential KPI requirements needed to support the use case include:

[P.R.5.4-001] The 5G system shall support “Remote control part” downloading for split control for robotics with a maximum latency as given in Table 5.4.6.1-1 (corresponding size of “Remote control part” is also listed in the table).

[P.R.5.4-002] The 5G system shall support “Remote control part” downloading for split control for robotics with a user experienced DL data rate as given in Table 5.4.6.1-1 (corresponding size of “Remote control part” is also listed in the table).

[P.R.5.4-003] The 5G system shall support “Remote control part” downloading for split control for robotics with communication service availability not lower than 99.999 %.

Table 5.4.6.1-1: Data rate and latency requirements for robotic control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Control mode** | **User experienced UL data rate for sensing data uploading** | **Size of “Remote control part” : maximum** | **User experienced DL data rate for “Remote control part” downloading** | **Latency for “Remote control part” downloading: maximum** |
| Full control at edge | 4.7Mbit/s | 200Bytes | 1.6Mbit/s | 1ms |
| Split control | 4.7Mbit/s | 40kBytes | 320Mbit/s | 12ms |

## 5.5 Session-specific model transfer split computation operations

### 5.5.1 Description

A UE, to achieve results for the user, employs split computation (The split computation is for offloading computation intensive task between UE and network). Computation intensive tasks (machine learning, complex computation using input data and the model, etc.) can be fully or partially offloaded. This use case considers a particular use – rendering augmented reality in a headset with modest computational resources. The decision how to split the computation task between the UE and other computation resources can depend on the conditions of the communication network and on computational resources available in the UE.

NOTE 1: The decision of how to split computation is itself out of scope of 3GPP and not discussed here.

### 5.5.2 Pre-conditions

Abigail has Augmented Reality glasses, a UE with limited computational power. She leaves a bus and stands at the bus stop, where, behind a large advertisement display, a gNB is installed. Abigail’s glasses get access through the access point. She seeks to augment her view of the city with directions and annotations (opening hours, local history, description of businesses, etc.) Augmenting the visual scene of the city in real time is a computationally intensive task, accomplished by a model developed through ML. the model has two candidate split points, each candidate split point has a different workload and communication requirement shown as below. This strategy for splitting computation has been installed in the UE and Application Server so that the split point can be adjusted dynamically based on change of communication performance and/or UE’s capabilities. The Application Server can be either in the MNO domain (i.e. a trusted application) or external to the MNO domain (i.e. an authorized third party application.) . This use case does not consider how the splitting strategy is determined, whether this can be done autonomously or what form the strategy has.

Table 5.5.2-1: Workload and communication requirement for split points

|  |  |  |
| --- | --- | --- |
|  | **Approximate output UL data rate (Mbit/s)** | **Computation load in UE** |
| Candidate split point 1 | 120 | Low |
| Candidate split point 2 | 24 | High |

The glasses have limited computational capacity, and this capacity varies over time. A means for identifying the current status (AI-ML information) of the UE is available to the network i.e. via application layer. Initially it is determined that the UE has the capability to support either candidate split point 1 or 2.

The network communication resources are enormous, it is determined by the augmented reality service to apply candidate split point 1 so that computation is executed mainly in the network, receiving large quantities of data provided by her glasses and this helps reduce computation in UE. The large quantity of data is transmitted via the QoS flow with guaranteed data rate (GBR) 200 Mbit/s.

### 5.5.3 Service Flows

**Case-a (split point adjusted based on communication performance):**

Abigail walks away from the bus stop and the vicinity of the hot spot.

As Abigail stood a few meters from the gNB hotspot, The insufficient communication resources leads to the serving gNB becoming unable to keep the QoS flow with GBR 200 Mbit/s anymore Thus the policy decision point (which could be anywhere – we leave out what takes the decision and how) determines to downgrade will be executed at the time point-a the GBR from 200 Mbit/s to 30 Mbit/s and immediately notifies UE and Application server of this downgrade. The strategy for splitting computation for the AR application now must be adjusted, i.e. change to candidate split point 2, for which more computation needs to be done locally but the required bit rate for UL transmission is reduced to 24 Mbit/s .

The strategy and constraints for the partition of work is out of scope of this use case. (These could include e.g. partial results could be sent to the UE, which could perform sub-optimally with reduced resources, can model information be sent in a lossy / compressed form that is still useful, etc.) In any case, one of the crucial inputs to the decision of how to split the work *is the current set of communication resources available.*

The network provides current network resource information concerning the UE to network communication performance such as new QoS parameters (GBR=20Mbit/s), the condition information (time point-a) for update of a new QoS, as well as end to end performance between the UE and the computation resources (e.g. in the Service Hosting Environment). This information is made available (exposed) to the split computation ‘policy decision point’ (which could be anywhere – in the UE, the edge, the cloud, etc., this is not relevant to the use case.)

**Case-b (split point adjusted based on UE’s AI-ML information):**

Originally candidate split point-2 is selected when UE’s capability can support a high work load.

Abigail’s UE’s computational information is monitored at the application layer. When the communication resources are sufficient to support Candidate split point 1, if the UE’s conditions degraded sufficiently (e.g. due to depleted battery, lack of storage, reduced computation capacity) then this would be a reason to select Candidate split point 1.

Then the split computation decision point then adjusts the split computation strategy:

**For case-a**: to avoid service interruption, the split computation decision point selects the new split point-2 before the time point-a arrives. How this is communicated or ‘enforced’ is out of scope of this use case and it is not suggested that this would be standardized.

**For case-b**: to guarantee the user experience, the split computation decision point selects the split point-1 as the UE’s status is insufficient to support a high work load anymore.

The UE status information is assumed to be collected via application layer.

### 5.5.4 Post-conditions

Abigail has no awareness of the change of model split point and continues to enjoy acceptable performance as she ventures into the city, even if perhaps it is not as good as when she stood at the bus stop. Note that this use case doesn’t conclude as long as Abigail continues to use the service – as the UE to network communication performance can change at any time.

### 5.5.5 Existing features partly or fully covering the use case functionality

From TS 22.261 v17.1.0, clause 6.6.2:

Based on operator policy, the 5G system shall support an efficient mechanism for selection of a content caching application (e.g. minimize utilization of radio, backhaul resources and/or application resource) for delivery of the cached content to the UE.

NOTE 1: The selection of content caching relies upon knowledge of communication resources (e.g. radio, backhaul, application) for delivery of content to the UE. Thus this requirement satisfies the requirements partly.

From TS 22.261 v17.1.0, clause 6.7.2:

The 5G system shall be able to provide the required QoS (e.g. reliability, end-to-end latency, and bandwidth) for a service and support prioritization of resources when necessary for that service.

The 5G system shall be able to support E2E (e.g. UE to UE) QoS for a service.

NOTE 2: E2E QoS needs to consider QoS in the access networks, backhaul, core network, and network to network interconnect.

The 5G system shall be able to support QoS for applications in a Service Hosting Environment.

From TS 22.261 v17.1.0, clause 6.8:

Based on operator policy, the 5G system shall support a real-time, dynamic, secure and efficient means for authorized entities (e.g. users, context aware network functionality) to modify the QoS and policy framework. Such modifications may have a variable duration.

Based on operator policy, the 5G system shall maintain a session when prioritization of that session changes in real time, provided that the new priority is above the threshold for maintaining the session.

From TS 22.261 v17.1.0, clause 6.10.2:

Based on operator policy, the 5G network shall provide suitable APIs to allow a trusted third-party application to request appropriate QoE from the network.

Based on operator policy, the 5G network shall expose a suitable API to an authorized third-party to provide the information regarding the availability status of a geographic location that is associated with that third-party.

Based on operator policy, the 5G network shall expose a suitable API to allow an authorized third-party to monitor the resource utilisation of the network service (radio access point and the transport network (front, backhaul)) that are associated with the third-party.

### 5.5.6 Potential New Requirements needed to support the use case

[P.R.5.5-001] Based on operator policy, the 5G network shall be able to provide the means to allow an authorized third-party to monitor the resource utilisation of the network service that are associated with the third-party.

[P.R.5.5-002] Based on operator policy, the 5G system shall be able to expose QoS information to an authorized 3rd party. The QoS information can include e.g. UE UL/DL bitrate, latency, reliability per location.

[P.R.5.5-003] The 5G system shall be able to provide the means to predict and expose network condition changes (e.g. bitrate, latency, reliability) to the authorized third party.

# 6 AI/ML model/data distribution and sharing over 5G system

## 6.1 AI/ML model distribution for image recognition

### 6.1.1 Description

Image recognition is an area where a rich set of pre-trained AI/ML models are available. The optimum model depends on the feature of the input image/video, environment and the precision requirement. The model used for vision processing at device needs to be adaptively updated for different vision objects, backgrounds, lighting conditions, purposes (e.g. image recovery vs. classification), and even target compression rates. Although a static model can also work as default in some cases, adapting the model to different working conditions will provide improved recognition accuracy and better user experience.

An example was given in [20] for the motivation of selecting the optimum model for different image recognition tasks and environments. As shown in Figure 6.1.1-1, 4 typical CNN models were evaluated and compared for different image recognition tasks, i.e. MobileNet\_v1\_025 [19], ResNet\_v1\_50 (ResNet [18] with 50 layers), Inception\_v2 [23], and ResNet\_v2\_152 (ResNet with 152 layers). This example shows that the best model depends on the type of input image and the task requirement. For a mobile device which needs to recognize diverse types of images and meet various requirements for different applications, the model needs to be adaptively switched.

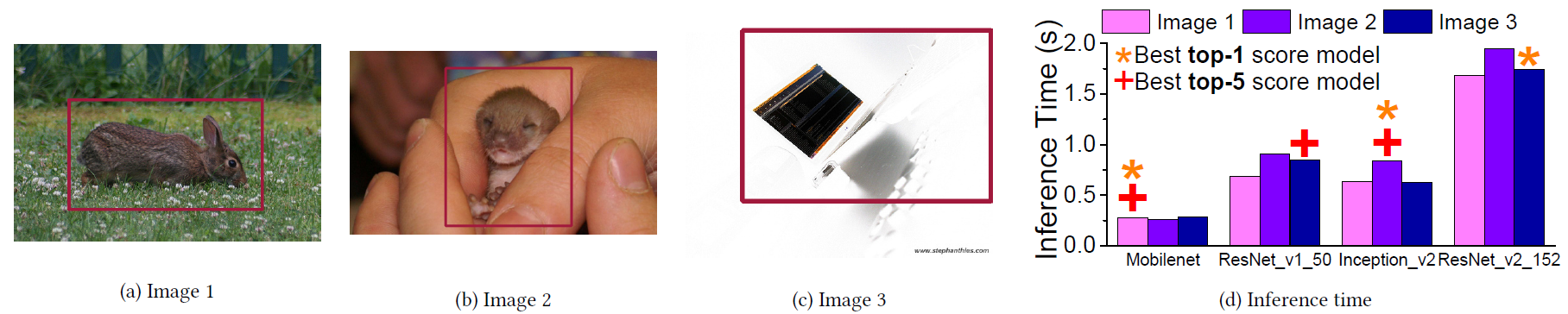


Figure 6.1.1-1. Example of selecting the optimum model for different image recognition tasks/environments (figure adopted from [20])

In case the selected model has not pre-loaded in the device, the device needs to download it from the network before the image recognition task can start. A model can be reused if it is kept in storage after the previous use. But due to the limited storage resource, the device cannot retain all models for potential use in storage. The data rate for downloading the needed models depends on the size of the model and the required downloading latency.

Along with the increasing performance requirements to AI/ML operations, the size of the models also keeps increasing, although model compression techniques are under improvements. The typical sizes of typical DNN models for image recognition are listed in Table 6.1.1-1. A DNN parameter can be expressed in 32 bits for a higher inference accuracy. The model size and downloading overhead can be compressed if the size of a parameter is reduced to 8 bits, by potentially sacrificing the image recognition accuracy.

The required model downloading latency depends on how fast the model needs to be ready at the device. It is impacted by the extent to which the oncoming application can be predicted. In the use case, we assume the device cannot predict and download the needed model in advance. In this case, the downloading of the AI/ML model needs to be finished in seconds or even in milliseconds. Different from a streamed video which can be played when a small portion is buffered, a DNN model can only be used until the whole model is completely downloaded.

For example, if the downloading latency is 1s, the required DL data rate ranges from 134.4 Mbit/s to 1.92Gbit/s in case of 32-bit parameters, as shown in Table 6.1.1-1. In case of 8-bit parameters, the required DL data rate can be limited to 33.6Mbit/s~1.1Gbit/s.

A model consists of model topology and model weight factors. The topology reflects the structure of a neural network (e.g. neurons of each layer, the connections of neurons between two neighboring layers). The size of the model parameters is shown in Table 6.1.1-1. The size of the configuration file of a model (i.e. the model topology), usually does not exceed 1Mbits. When an application of the UE requests a model, the third-party server sends one model profile which may consist of two parts, i.e. model topology and model weight factors. Because the model topology and model weight factors are from the same server, probably they have the same IP tuple. The transmission error of model topology matters a lot comparing to model weight factors (the UE can hardly run the model if model topology has an error while the weight factors may have a high transmission error tolerance due to model robustness). Therefore, the transmission of data of model topology is more important and requires higher reliability.

Table 6.1.1-1: Sizes of typical image-recognition models and required DL data rates for downloading in 1s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DNN model for image recognition** | **Number of parameters (Million)** | **32 bits per parameter** | | **8 bits per parameter** | |
| **Size of the model (MByte)** | **Required DL data rate (Mbit/s)** | **Size of the model (MByte)** | **Required DL data rate (Mbit/s)** |
| AlexNet [7] | 60 | 240 | 1920 | 60 | 480 |
| VGG16 [8] | 138 | 552 | 4416 | 138 | 1104 |
| ResNet-152 [18] | 60 | 240 | 1920 | 60 | 480 |
| ResNet-50 [18] | 25 | 100 | 800 | 25 | 200 |
| GoogleNet [9] | 6.8 | 27.2 | 217.6 | 6.8 | 54.4 |
| Inception-V3 [23] | 23 | 92 | 736 | 23 | 184 |
| 1.0 MobileNet-224 [19] | 4.2 | 16.8 | 134.4 | 4.2 | 33.6 |

### 6.1.2 Pre-conditions

The UE runs an application providing the capability of AI/ML model inference for image recognition.

An AI/ML server manages the AI/ML model pool, and is capable to download the requested model to the application providing AI/ML based image recognition.

The 5G system has the ability to provide 5G network related information to the AI/ML server.

### 6.1.3 Service Flows

1. The AI/ML based image recognition application is requested by the user to start recognizing the image/video shot by the UE.
2. The AI/ML model is downloaded from the model server to the AI/ML based image recognition application via 5G network.
3. The AI/ML based image recognition application employs the AI/ML model for inference until the image recognition task is finished.
4. Redo Step 2) to 3) for AI/ML model re-selection and re-downloading if needed to adapt to the changing conditions.

### 6.1.4 Post-conditions

The objects in the input images or videos are recognized by the AI/ML based image recognition application and the inference accuracy and latency need to be guaranteed.

The image recognition task can be completed under the available computation and energy resource of the UE.

### 6.1.5 Existing features partly or fully covering the use case functionality

This use case mainly requires high data rate together with low latency. The high data rate requirements to 5G system are listed in Clause 7.1 and 7.6 of TS22.261 [4]. As in Table 7.1-1 of [4], 300Mbps DL experienced data rate and 50Mbps UL experienced data rate are required in dense urban scenario, and 1Gbps DL experienced data rate and 500Mbps UL experienced data rate are required in indoor hotspot scenario. As in Table 7.6.1-1 of [4], cloud/edge/split rendering-related data transmission requires up to 0.1Gbps data rate with [5-10]ms latency countrywide.

### 6.1.6 Potential New Requirements needed to support the use case

Considering the time taken by the device to finish the image recognition task, a small portion of the recognition latency budget can be used to download the model. For one-shot object recognition at smartphone and photo enhancements at smartphone, a pre-installed sub-optimal model can be temporarily employed while downloading the optimal model. The optimal model in need should be downloaded on level of 1s, and then replaces the pre-installed model. If 8-bit parameters are used for describing the DNN, the required DL data rate ranges from 33.6Mbit/s to 1.1Gbit/s. For video recognition, the target can be updating the model in one frame duration (so to adopt the updated model for the next frame). But for an application with an always-on camera, the device can predict the needed model and start the downloading in-advance. Downloading the model within 1s is acceptable. Similarly, for other applications with an always-on camera, i.e. person identification in security surveillance system, AR display/gaming, remote driving, remote-controlled robotics, the required DL data rate ranges from 33.6Mbit/s to 1.1Gbit/s. It should be noted that the size of the model may be further reduced if more advanced model compression techniques can be adopted.

#### 6.1.6.1 Potential KPI Requirements

The potential KPI requirements needed to support the use case include:

[P.R.6.1-001] The 5G system shall support AI/ML model downloading for image recognition with a maximum latency as given in Table 6.1.6.1-1.

[P.R.6.1-002] The 5G system shall support AI/ML model downloading for image recognition with a user experienced DL data rate as given in Table 6.1.6.1-1.

[P.R.6.1-003] The 5G system shall support AI/ML model downloading for image recognition with communication service availability not lower than 99.999 %.

[P.R.6.1-004] The 5G system shall support AI/ML model downloading with a reliability for the transmission of data of model topology not lower than 99.999% and a reliability for the transmission of data of model weight factors not lower than 99.9%.

Table 6.1.6.1-1: Image recognition model downloading latency analysis for example applications (8-bit parameters for the DNN)

|  |  |  |  |
| --- | --- | --- | --- |
| **User application** | **Latency: maximum** | | **User experienced DL data rate for model downloading** |
| **Image recognition latency** | **Model downloading latency** |
| One-shot object recognition at smartphone | ~1s | 1s (Note 1) | 33.6Mbit/s~1.1Gbit/s |
| Person identification in security surveillance system | ~1s | 1s (Note 2) | 33.6Mbit/s~1.1Gbit/s |
| Photo enhancements at smartphone | ~1s | 1s (Note 1) | 33.6Mbit/s~1.1Gbit/s |
| Video recognition | 33ms@30FPS | 1s (Note 2) | 33.6Mbit/s~1.1Gbit/s |
| AR display/gaming | <5ms (Note 3) | 1s (Note 2) | 33.6Mbit/s~1.1Gbit/s |
| Remote driving | <5ms (Note 4) | 1s (Note 2) | 33.6Mbit/s~1.1Gbit/s |
| Remote-controlled robotics | <5ms (Note 5) | 1s (Note 2) | 33.6Mbit/s~1.1Gbit/s |
| NOTE 1: A pre-installed sub-optimal model can be temporarily employed while downloading the optimal model. The optimal model should be downloaded in the order of 1s.  NOTE 2: For applications with an always-on camera, the device can predict the needed model and start the downloading in-advance. Downloading the model within 1s is acceptable.  NOTE 3: According to [4][5], the VR motion-to-photon latency is in the range of 7-15ms. It can be assumed that the AR display or gaming requires a similar end-to-end latency. Considering the time taken by the AR rendering (e.g. 3D rendering of the virtual objects, rendering augmentation in overlay), the background video recognition should be finished within 5ms.  NOTE 4: According to [46] the one-way latency required for remote driving is 5ms. The round-trip latency is assumed to be 10ms. Considering the time taken by the vision-based driving inference at server, the latency budget for traffic object recognition can be estimated to be 5ms.  NOTE 5: According to [5] the end-to-end latency required for video-operated remote-controlled robotics is 10~100ms. Considering the time taken by the robot controlling inference at server, the robot vision recognition needs to be finished within 5ms. | | | |

## 6.2 Real time media editing with on-board AI inference

### 6.2.1 Description

Smartphone is the #1 device that people carry and use everywhere for audio and video recording. It also becomes the first device to exchange media content with friends and family, to publish on social media. High end smartphones embed more and more powerful CPU and GPU and even dedicated AI hardware accelerators. As camera and picture/video quality become a differentiator among high end smartphones, AI/ML models to enhance photo shoots locally emerge on these high-end devices. AI accelerators are expected to enable the execution of complex AI/ML models directly on end-users connected devices; not only photo enhancements but high-quality audio and video content analysis and enhancement are expected to be executed locally on smartphones. Smartphones will consequently become a device to edit media content prior to sharing over the network. With the advent of 5G, new services relying on on-demand downloads of large AI/ML models to be executed in (near) real time on end user device will emerge; depending on the service, the environment, the user’s preference, the device characteristics, etc., these DNN models will need to be adapted or updated under strict latency constraints which prevent all of them to be stored locally in advance.

Media content analysis combines tasks as object detection, segmentation, face recognition, people counting, human activity tracking. In table 6.2.1-1, examples of commonly used DNN models for object detection (with their respective sizes) are listed.

Table 6.2.1-1: Sizes of typical object detection models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model for object detection** | **Number of parameters (Million)** | **Size of the model (MByte)  32 bits parameters** | **Size of the model (MByte) 8 bit parameters** |
| MobileNet | 3.2 | 12.8 | 3.2 |
| DarkNet | 20 | 80 | 20 |
| SE ResNet | 26 | 104 | 26 |
| Inception v4 | 41 | 164 | 41 |
| YOLONet | 64 | 256 | 64 |
| VGGNet | 134 | 536 | 134 |

Media content edition combines tasks as audio and video quality improvement, language translation, face anonymisation. In tables 6.2.1-2 and 6.2.1-3, examples of commonly used DNN models for image super-resolution and for video super-resolution are listed.

Table 6.2.1-2: Sizes of typical image super-resolution models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model for image super-resolution** | **Number of parameters (Million)** | **Size of the model (MByte)  32 bits parameters** | **Size of the model (MByte) 8 bits parameters** |
| RCAN | 15.44 | 61.78 | 15.44 |
| SAN | 15.71 | 62.82 | 15.71 |
| RDN | 22.12 | 88.48 | 22.12 |
| EDSR | 40.73 | 162.92 | 40.73 |
| OISR-RK3 | 41.91 | 167.64 | 41.91 |

Table 6.2.1-3: Sizes of typical video super-resolution models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model for video super-resolution** | **Number of parameters (Million)** | **Size of the model (MByte)  32 bits parameters** | **Size of the model (MByte) 8 bits parameters** |
| RBPN/4-PF | 12.7 | 50.8 | 12.7 |
| RBPN/6-PF | 12.7 | 50.8 | 12.7 |
| VSR-DUF | 6.8 | 27.2 | 6.8 |
| DRDVSR | 0.7 | 2.8 | 0.7 |

Two settings are considered for the use case:

1. Independent user: a person takes a video or starts a video call on its UE in a noisy environment, with difficult lighting conditions, and automatic tagging of scene and objects are embedded in the video.
2. Crowd: During a large event, like a live concert, several thousand people use their UEs to film or photograph the musician band at the same time, and request additional information on the concert like band discography, lyrics, artist facial recognition, instrument/equipment brand. In this context, UEs request the downloads of DNN models to improve the capture and recording of the concert, and to provide information requested by people attending the concert. Several DNN models can be requested by each UE to e.g. execute the following tasks: image shooting and video optimization, artist face recognition, musical instrument identification, audio improvement and lyrics generation. Given the heterogeneous fleet of UEs, thousands of DNN models required by the application/service – can be requested for download. These DNN models are adapted or updated to the UEs operating system type and version, hardware characteristics and environment.

### 6.2.2 Pre-conditions

The setting for this use case is as follows. Alice is attending a crowded live concert. She is eager to get movie clips and pictures as a great souvenir of the concert, but she is worried about difficult conditions to get this great souvenir as the conditions for light and sound are very variable, with limited light for the spectators and too much light on the scene. The audio stereo is variable and not well-balanced dependent on where she is among the audience and has a very noisy background.

Alice would like to store good quality movie clips and pictures on her private account on the internet for the future, and also post photos and videos tagged with artist name and other relevant information during the concert. As Alice is also an amateur musician, she also wants to get detailed real-time information about the structure of the song, the lyrics and the instruments.

The pre-conditions are:

1. Alice is attending a crowded live concert.
2. Her UE is registered to the 5G network.
3. Applications of Alice’s smartphone can rely on fine-tuned machine learning models that are available via the network that covers the concert hall. These models are:
4. An ML model improving photo capture for this concert hall (a model specially fine-tuned for this concert place).
5. An ML model improving audio capture for this concert hall (a model specially fine-tuned for this concert place).
6. An ML model specialized in the discography and the lyrics of the band.
7. An ML model specialized in the artists face recognition.
8. An ML model specialized in music instrument identification.

NOTE: The listed models above are examples for this use-case and is not an exhaustive list.

### 6.2.3 Service Flows

1) Shortly after the start of the concert, Alice, as most of the fans, launches the camera application on her mobile phone to film the scene and to get additional information about the band, the individual artists or the songs or instruments.

2) She points her device’s camera towards the scene.

3) The environment is very dark with strong light spots. To be fully functional and to render the best user experience, the camera application downloads ad-hoc ML models.

4) The proposed ML models are very performant in this environment but also very heavy in size.

5) The ML models can be used for the whole capture especially if environment remains stable. If the environment changes substantially, or better ML models become available, ML models can be progressively updated accordingly with respect of some operating rules (like a maximum number of ML update per second). In all cases, the camera application continues working seamlessly.

6) The audio and video streams are captured, improved in quality, processed to extract and display additional information, and either stored in real-time on the mobile phone itself or provided as a live stream.

### 6.2.4 Post-conditions

Alice can see that even in harsh light conditions and with the noisy background the photos and videos are great, additional information is provided and all is correctly tagged as requested.

The post-conditions are:

1. Photos and videos are either stored on the mobile phone in the improved high quality, ready to be uploaded and shared on social media, or directly shared on social media.
2. Audio recording is high quality with ambient noise reduction, improved stereo balance.
3. Additional information about band, song/lyrics, instruments, etc. are displayed on the mobile phone and stored in media recordings’ metadata.
4. Alice can visualize additional information and upload the photos and videos on her social network(s) with the associated tags and information provided by the models, and also store the above on her personal media server.

### 6.2.5 Existing features partly or fully covering the use case functionality

The performance requirements for high data rate and traffic density scenarios are found in TS 22.261 [4] clause 7.1. The scenario Broadband access in a crowd is relevant for the use case of very dense crowds, for example at stadiums or concerts. In addition to a very high connection density, the users can share their experience, i.e. what they see and hear. This can put a higher requirement on the uplink than the downlink.

This new use case has some requirements on the downlink not covered by the existing requirements, see table 6.2.5-1.

Table 6.2.5-1. Excerpt from TS 22.261 [4] Table 7.1-1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Scenario | Experienced data rate (DL) | Experienced data rate (UL) | Area traffic capacity  (DL) | Area traffic capacity  (UL) | Overall user density | Activity factor | UE speed | Coverage |
| 4 | Broadband access in a crowd | 25 Mbit/s | 50 Mbit/s | [3,75] Tbit/s/km2 | [7,5] Tbit/s/km2 | [500 000]/km2 | 30% | Pedestrians | Confined area |

### 6.2.6 Potential New Requirements needed to support the use case

#### 6.2.6.1 Introduction

Potential new requirements needed to support the use case result of the following parameters:

1. AI/ML model size.
2. Accuracy of the model.
3. latency constraint of the application or service.
4. number of concurrent downloads, i.e. number of UEs requesting AI/ML model downloads within same time window.

The number of concurrent downloads further depends on the density of UE in the covered area and the covered area size.

The tables 6.2.6.1-1, 6.2.6.1-2 and 6.2.6.1-3 contain KPI for different aspects of the real-time media editing use case.

Table 6.2.6.1-1. Typical sizes of AI/ML models for the UC

|  |  |  |  |
| --- | --- | --- | --- |
| **AI/ML Model** | **Number of parameters (Million)** | **Size of the AI/ML model (MByte)** | **Comments** |
| MobileNet | 3.2 | 3.2 | 8-bit parameters |
| MobileNet | 3.2 | 12.8 | 32-bit parameters |
| RCAN | 15.44 | 15.44 | 8-bit parameters |
| DarkNet | 20 | 20 | 8-bit parameters |
| Inception v4 | 41 | 41 | 8-bit parameters |
| RCAN | 15.44 | 61.78 | 32-bit parameters |
| YOLONet | 64 | 64 | 8-bit parameters |
| DarkNet | 20 | 80 | 32-bit parameters |
| VGGNet | 134 | 134 | 8-bit parameters |
| Inception v4 | 41 | 164 | 32-bit parameters |
| YOLONet | 64 | 256 | 32-bit parameters |
| VGGNet | 134 | 536 | 32-bit parameters |

From table 6.2.6.1-1, AI/ML models currently available to elaborate the use case have sizes that vary from 3.2 MB to 536 MB.

As indicated in the use case in clause 6.1, it can be noted that the size of AI/ML models can be reduced prior to transmission with dedicated model compression techniques. On the contrary, AI/ML models with more neural network layers and more complex architectures arise to solve more complex tasks and to improve accuracy. This trend is expected to continue in the coming years. Typical model sizes in the range of 3 MB to 500 MB appear to be a reasonable compromise to consider for this use case.

In the following, two categories are considered for AI/ML model sizes:

1. AI/ML model sizes below 64 MB, which can be associated to models optimized for fast transmission,
2. AI/ML model sizes below 500 MB, which can be associated to models optimized for higher accuracy.

Maximum latency in function of application or service:

1. Videocall service: end-to-end latency below 200 ms,
2. Video recording, video streaming, and object recognition applications: latency below 1 s.

User experienced DL data rate results from above AI/ML models’ sizes and maximum latency values are summarized in the table 6.2.6.1-2.

Table 6.2.6.1-2. UC AI/ML model download – single UE – KPIs

|  |  |  |  |
| --- | --- | --- | --- |
| **UC model download** | **AI/ML model size** | **Latency: maximum** | **User experienced DL data rate** |
| Single Model / Single UE | [3 MB – 64 MB] | <1 s | [24 Mb/s ~ 512 Mb/s] |
| Single Model / Single UE | [3 MB – 64 MB] | <200 ms | [120 Mb/s ~ 2.56 Gb/s] |
| Single Model / Single UE | [64 MB – 500 MB] | <1 s | [512 Mb/s ~ 4 Gb/s] |
| Single Model / Single UE | [64 MB – 500 MB] | <200 ms | [2.56 Gb/s ~ 20 Gb/s] |

As indicated above, the number of concurrent downloads is a third parameter to determine potential new requirements. This corresponds to the maximum number of UEs requesting a AI/ML model download in a same time window and same covered area/cell.

The case of a concert hall is an illustration of the scenario, “Broadband access in a crowd” from TS 22.261 [4]. This scenario assumes an overall user density of 500 000 UE / km2 (i.e. 0.5 UE / m2) and an activity factor of 30 %.

In the concert hall case, it is also assumed that only a part of the UEs intends to request AI/ML model downloads. Moreover, only a subpart will request AI/ML model download during the same time window in the same cell. The activity factor is finally estimated to 1 % (i.e. % of UE requesting an AI/ML model download within same time window).

Typical number of UEs in a concert hall varies from ~1000 seats to ~ 5000 seats.

Based on these figures and UE activity assumption, the number of concurrent downloads is estimated as given in table 6.2.6.1-3.

Table 6.2.6.1-3. Estimated number of concurrent downloads

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of UEs** | **Estimated area**  **(density of**  **0.5 UE / m2 )** | **Activity Factor** | **Number of concurrent downloads in the same cell** |
| 1000 | 2000 m2 | 1 % | 10 |
| 5000 | 10000 m2 | 1 % | 50 |

From table 6.2.6.1-2 and table 6.2.6.1-3, requirements on the covered area are estimated as follows:

Table 6.2.6.1-4. Estimated covered area DL data rate requirements

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of UEs** | **Activity Factor** | **Number of concurrent downloads** | **AI/ML model size** | **Latency: maximum** | **User experienced DL data rate** | **Aggregated user experience DL data rate for covered area** |
| 1000 | 1 % | 10 | [3 MB – 64 MB] | <1 s | [24 Mb/s ~ 512 Mb/s] | [240 Mb/s ~ 5.12 Gb/s] |
| [3 MB – 64 MB] | <200 ms | [120 Mb/s ~ 2.56 Gb/s] | [1.2 Gb/s ~ 25.6 Gb/s] |
| [64 MB – 500 MB] | <1 s | [512 Mb/s ~ 4 Gb/s] | [5.12 Gb/s ~ 40 Gb/s] |
| 5000 | 1 % | 50 | [3 MB – 64 MB] | <1 s | [24 Mb/s ~ 512 Mb/s] | [1.2 Gb/s ~ 25.6 Gb/s] |
| [3 MB – 64 MB] | <200 ms | [120 Mb/s ~ 2.56 Gb/s] | [6 Gb/s ~ 128 Gb/s] |
| [64 MB – 500 MB] | <1 s | [512 Mb/s ~ 4 Gb/s] | [25.6 Gb/s ~ 200 Gb/s] |

Another approach to estimate the number of concurrent downloads is to estimate the number of different AI/ML models requested by UEs instead of the number of UEs requesting AI/ML models. The AI/ML models can then be broadcast/multicast to multiple UEs. The number of different AI/ML models depends on the accuracy expectations of the AI/ML models, the execution environments and the hardware characteristics of end devices. When the number of UE requesting AI/ML models is very high, the number of different AI/ML models can remain smaller. This approach is well suited for very large crowd.

The number of concurrent downloads when transmitted in broadcast/multicast to many UEs can be estimated between 1 (i.e. all UEs request the same AI/ML model) and 50 (i.e. all UEs request different AI/ML models).

#### 6.2.6.2 Potential KPI Requirements

Independent user:

[P.R.6.2-I-001] The 5G system shall support the download of AI/ML models with a latency below 1s and a user experienced data rate of 512 Mb/s.

NOTE 1: This requirement concerns AI/ML models having a size below 64 MB.

[P.R.6.2-I-002] The 5G system shall support the download of AI/ML models with a latency below 1 s and a user experienced data rate of 4 Gb/s.

NOTE 2: This requirement concerns AI/ML models having a size below 500 MB.

Crowd:

[P.R.6.2-C-001] The 5G system shall support the parallel download of up to 50 AI/ML models with a latency below 1 s.

NOTE 3: This requirement concerns AI/ML models having a size below 64 MB.

[P.R.6.2-C-002] The 5G system should support the functionality to broadcast/multicast a same AI/ML model to many UEs with a latency below 1 s.

NOTE 4: This requirement concerns AI/ML models having a size below 64 MB.

## 6.3 AI/ML model distribution for speech recognition

### 6.3.1 Description

AI/ML-based speech processing has been widely used in applications on mobile devices (e.g. smartphone, personal assistant, language translator), including automatic speech recognition (ASR), voice translation, speech synthesis. Speech recognition for dictation, search, and voice commands has become a standard feature on smartphones and wearable devices.

Service requirements to ASR have been addressed in [29]. Traditional ASR systems are based on hidden Markov model (HMM) and Gaussian mixture model (GMM). However, the HMM-GMM systems suffer a relatively high WER (Word Error Rate) with presence of environmental noise. Although some enhancements were developed including “feature enhancement” (attempts to remove the corrupting noise from the observations prior to recognition) and “model adaptation” (leaves the observations unchanged and instead updates the model parameters of the recognizer to be more representative of the observed speech), the traditional models can hardly fulfil the requirements of commercial applications. Acoustic models based on deep neural networks (DNN) have remarkable noise robustness [25-26], and have been widely used in the ASR applications in mobile devices.

Nowadays, most of ASR applications on smartphones are operating in cloud servers. The end device uploads the speech to the cloud server, and then downloads the decoded results back to the device. However, cloud-based speech recognition potentially introduces a higher latency (not only due to the 4G/5G network latency, but also the internet latency), and the reliability network connection and privacy issue need to be considered.

An embedded speech recognition system running on a mobile device is more reliable and can have lower latency. Currently, some ASR apps would switch from cloud-based model inference to offline model inference when uplink coverage of the mobile user turns weak, e.g. when entering a basement or an elevator. However, the ASR models for cloud servers are too complex for computation and storage resources on mobile devices. The size of a ML-based ASR model running on cloud server has rapidly increased in the recent year, from ~1GByte to ~10GByte, which cannot be run on a mobile device. Due to the restriction, only the simple ASR applications, e.g. wakeword detection, can be implemented on smart phones. Realizing more complicated ASR applications, e.g. large vocabulary continuous speech recognition (LVCSR) is still a challenging area for an offline speech recognizer.

In 2019, a state-of-the-art offline LVCSR recognizer for Android mobile devices was announced. The streaming end-to-end recognizer is based on the recurrent neural network transducer (RNN-T) model [27-28]. It was stated that, by employing all kinds of improvements and optimizations, the memory footprint can be dramatically reduced and the computation can be speed up. The model can be compressed to 80MB. Meanwhile the ASR model is compressed to fit the use at mobile device, the robustness to the various types of background noises has to been sacrificed. When the noise environment changes, the model needs to be re-selected, and in case the model is not kept in the device, the model needs to be downloaded from the cloud/edge server of the AI/ML model owner via 5G network.

### 6.3.2 Pre-conditions

The UE runs an application providing the capability of AI/ML model inference for speech recognition.

An AI/ML server manages the AI/ML model pool, and is capable to download the requested model to the application providing AI/ML based speech recognition.

The 5G system has the ability to provide 5G network related information to the AI/ML server.

### 6.3.3 Service Flows

1. The AI/ML based speech recognition application is requested by the user to start recognizing the speech recorded.
2. The AI/ML model is downloaded from the model server to the AI/ML based speech recognition application via 5G network.
3. The AI/ML based speech recognition application employs the AI/ML model for inference until the speech recognition task is finished.
4. Redo Step 2) to 3) for AI/ML model re-selection and re-downloading if needed to adapt to the changing conditions.

### 6.3.4 Post-conditions

The content in the input speech is recognized by the AI/ML based speech recognition application and the inference accuracy and latency need to be guaranteed.

The speech recognition task can be completed under the available computation and energy resource of the UE.

### 6.3.5 Existing features partly or fully covering the use case functionality

This use case mainly requires high data rate together with low latency. The high data rate requirements to 5G system are listed in Clause 7.1 and 7.6 of TS22.261 [4]. As in Table 7.1-1 of [4], 300Mbps DL experienced data rate and 50Mbps UL experienced data rate are required in dense urban scenario, and 1Gbps DL experienced data rate and 500Mbps UL experienced data rate are required in indoor hotspot scenario. As in Table 7.6.1-1 of [4], cloud/edge/split rendering-related data transmission requires up to 0.1Gbps data rate with [5-10]ms latency countrywide.

### 6.3.6 Potential New Requirements needed to support the use case

An ASR model can only be used until the whole model is completely downloaded. And device needs to adopt the noise-robust ASR model which adapts to the changing noise environment. In general, the device’s microphone is switched on only when the speech recognition application is triggered. The device needs to identify the noise environment and download the corresponding ASR model in a low latency (level of 1s). The sizes of some typical ASR models are listed [27][47] in Table 6.3.6-1, which are used to derive the required data rate.

#### 6.3.6.1 Potential KPI Requirements

The potential KPI requirements needed to support the use case include:

[P.R.6.3-001] The 5G system shall support AI/ML model downloading for speech recognition with a latency not higher than 1s.

[P.R.6.3-002] The 5G system shall support AI/ML model downloading for speech recognition with a user experienced DL data rate as given in Table 6.3.6.1-1.

[P.R.6.3-003] The 5G system shall support AI/ML model downloading for speech recognition with communication service availability not lower than 99.999 %.

Table 6.3.6-1: Sizes of typical speech-recognition models and user experienced DL data rates for downloading in 1s (8 bits per parameter)

|  |  |  |  |
| --- | --- | --- | --- |
| **DNN model for speech recognition** | **Number of parameters (Million)** | **Size of the model (MByte)** | **User experienced DL data rate (Mbit/s)** |
| RNN-CTC [47] | 26.5 | 26.5 | 212 |
| ResCNN-LAS [47] | 6.6 | 6.6 | 52.8 |
| QuartzNet-15×5 [47] | 19 | 19 | 152 |
| Gboard speech recognizer [27] | N.A. | 80 | 640 |

## 6.4 AI model management as a Service

### 6.4.1 Description

The AI/ML models can be classified and generated from different perspectives, such as services, users, time, locations, etc. So there can be a large number of AI/ML models (incl. data for structure and weight factors, gradient) used for inference and training. Due to the limitation of storage, UE cannot always preload all AI/ML models for different works. It can be a new commercial opportunity that operators provide services to help manage and distribute the AI/ML models so that UE can acquire a proper model immediately.

In the future, it is expected that 3rd party companies will make use of AI models to support different kinds of services such as panorama tourist guide using Augmented Reality (AR) in a resort. However, most of 3rd party does not have the resource (server) in distributed places. Considering the large size of AI models, strict downloading time (illustrated in use case 6.1, 6.2, 6.3) and limited UE storage, it is appropriate for 3rd party companies to authorize professional companies to manage their AI models rather than doing the same thing locally by themselves.

Since operator cloud resource has advantage to manage different AI/ML models centrally (e.g. cloud server) and locally (e.g. MEC server), such services can be exposed to 3rd party users as well. Specifically,

* The operator has its own need to maintain a AI/ML model pool for its own business like network optimization;
* The 3rd party would like to use some common AI/ML models already stored in operator’s cloud;
* Due to resource limitation of 3rd party, they would like to lease operator’s cloud resource to manage their AI models.

Therefore, as shown in Fig.6.4.1-1, The AI model management includes:

* Let 3rd party invoke capabilities exposed by 5GS to upload/download/update/delete/store/monitor AI models.
* Transmitting AI models to user efficiently per situation (e.g. entering into a certain area)
* As the 5GS may collect communication data, user experience, etc., it may perform a data analytics for user experience, which can generate analyzing results to help operator’s cloud train and improve AI models for the 3rd party.



**Figure 6.4.1-1: Operator cloud for model management**

### 6.4.2 Pre-conditions

The operator cloud (e.g. Edge server) stores a variety of AI/ML models according to operator’s or 3rd party’s requirement.

The operator cloud (e.g. Edge server) is capable to distribute a model stored in the operator cloud to devices.

### 6.4.3 Service Flows

* + - 1. Company A provides panorama tourist guide using Augmented Reality (AR) technology. They provide guide services in a commercial area and a tourist resort. UE needs to download Model A and B (both are 32 bits VGGnet, 536MByte) respectively;
      2. To reach the SLA, company A indicates to the operator that UE requires model A in area A and model B in area B.
      3. When UE moves to one place, The local Edge server trigger to establish a QoS acceleration and UE downloads the corresponding Model (A or B) stored in the local Edge server timely so that user can enjoy the continuous AR world without obvious interruption when model changes.

### 6.4.4 Post-conditions

UE uses the Model for panorama tourist guide.

### 6.4.5 Existing features partly or fully covering the use case functionality

None.

### 6.4.6 Potential New Requirements needed to support the use case

[P.R.6.4-001] Subject to user consent, operator policies and the regional or national regulatory requirements, the 5G system shall be able to provide the capability to expose information (e.g. measured data rate, delay, network analytics results) to an authorized third-party application to support the training and monitoring of the AI/ML models.

[P.R.6.4-002] The 5G system shall be able to support an authorized third-party application to distribute an AI/ML model ranging from 3.2~536MB to an third-party application running on the device in less than 1 second with a user density of up to 5000~10000/km2 in an urban area.

## 6.5 AI/ML based Automotive Networked Systems

### 6.5.1 Description

This use-case is illustrative for a specific example but is readily extensible for more advanced scenarios involving disaster scenarios, rescue operations or even autonomous vehicle operations. The focus of this scenario is on machine-to-machine AI/ML transfer learning and inference systems that are networked together using 5G networks to provide smart automotive applications and services.

The scenario assumes that there exist multiple AI/ML systems available, in a high-reliability, low-latency, high-bandwidth system optimized for machine-to-machine interaction. The proposed AI/ML systems may continuously exchange and share AI-ML model layers in a distributed and or federated network as determined by the system in response to a change of events, conditions or emergency situations, to improve some or all of the ML system prediction accuracy. These systems may optimize the AI/ML inference latency by executing different layers on various AI/ML networked systems.

There are two main types of ML models and model processing considered in this use-case:

1. Large ML models updates using non-real time training – These ML models are trained and optimized with millions or billions of parameters using extensive computing resources to achieve the highest accuracy possible based on specific sets of input training data. This training is performed over a long period of time and the resulting fully-trained ML model is the baseline model installed at initial production for the devices in the use-cases described below. These fully trained models may be updated with externally provided AI-ML model data and may also improve themselves based on external sensor data.
2. Partial transfer and exchange of AI-ML model data – This use case describes different applications where different types of ML systems are networked to exchange parts of AI-ML model data to improve prediction accuracy. In some systems, local ML models can continuously improve their baseline model using data gathered from the existing environment and other sensor input. These systems upload improved model data at relatively slow rates to a larger cloud-based network where further processing takes place to further refine the full ML model. The specific method used for continuous improvement is outside the scope of this use case.

Different example scenarios are described below.

**AI/ML Systems Emergency Response to Disabled Vehicle**

Figure 6.5.1-1 illustrates a vehicle that has broken down hidden by a blind curve in the road. This breakdown is considered high-severity for the purposes of illustrating this use-case.



Figure 6.5.1-1: Dangerous vehicle break-down hidden by a blind curve

On-board vehicle sensors detect the following: mechanical failure, speed reduction, GPS coordinates and a variety of technical parameters are used by on-board ML inferencing systems to immediately diagnose the failure and notify surrounding networks of the severity of the failure.

Different non-real time AI models can use the gathered failure training data to refine prediction and prevention models updates.

**Traffic Surveillance Camera and MEC Metro Traffic Safety System**

Surveillance cameras use ML models to analyze image sensor data to detect and categorize the type of accident using ML inference with continuous AI-ML model improvement. This processed accident information is uploaded along with the model updates to the local near-edge MEC for AI/ML system model prediction and algorithm processing to suggest the immediate next course of action. These AI/ML systems suggest an automated warning broadcast. The automated AI/ML systems relying on trained ML models to predict the position of the vehicle is especially dangerous due to the hidden blind curve and informs local Emergency Response Unit dispatch to send help to set up a traffic bypass earlier on the road. The local near-edge MEC models are capable of continuous model improvement and return this model data back to the surveillance camera for further inference accuracy improvement.

The uploading of data, possibly pre-processed by the local AI/ML model, from the surveillance camera to the MEC may happen with low-latency depending on the severity of the accident while returned data to update the local model may happen on a much slower timeframe.

**Approaching Intelligent Vehicle Reaction**



Figure 6.5.1-2: Approaching Autonomous Vehicle

Figure 6.5.1-2 illustrates a vehicle rapidly approaching the car illustrated in Figure 1. The vehicle’s sensors are unable to detect the hazard and must rely upon other AI/ML systems for a warning.

An approaching autonomous vehicle receives multiple warning notifications and AI/ML systems in the vehicle use trained models predict and suggest appropriate automatic manoeuvres to move the vehicle to the far lane to avoid any danger. Sharing of data with other systems allows for constantly refining their models; this sharing of data does not have non-real requirements.

### 6.5.2 Pre-conditions

The scenarios described above assume the following pre-conditions:

* All existing AI/ML systems have existing pre-trained default AI-ML models that will perform predictions at a baseline accuracy and are capable of updating/re-fining the AI-ML models for continuously improving accuracy for the specific purpose.

Table 6.5.2-1: Use-Case Pre-Conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Broken Autonomous Vehicle** | **Approaching Autonomous Vehicle** | **Road Surveillance Camera** | **Smart City Metro Traffic Control** | **Smart City Emergency Services Network** |
| AI/ML Vehicle model has detected the fault. All comm. systems are available & active. | AI/ML Vehicle models are all functioning properly. All comms systems ready &active. | Camera is powered, active and connected to the network. | Software and comms are running properly. | Software, Comms and emergency assets are running properly. |

### 6.5.3 Service Flows

Appropriate AI/ML systems shall provide security and data protection as dictated by law and other appropriate policies.

The service flows described in Figure 3 are all machine-to-machine AI/ML intelligent system interactions designed for the sharing of information to with a range of demands from lowest-latency for critical situations (e.g. alarm messages) and relaxed latency for less critical problems. Shared information (possibly including an extensive set of data) can also be exchanged with relaxed latency to support training and updating of the AI/ML model. Each level of service may be owned and operated by a different organization.

This use-case assumes that no more than [10%] of the total ML model size needs to be shared or to updated in a single-low latency transmission.



Figure 6.5.3-1: Examples of AI-ML interactions

Vehicle to other AI Systems

Examples of AI-ML model data exchange are:

* Vehicle health and inspection monitoring ML models – Local ML models sense and react to crash, fire, temperature and electrical sensor information. These models are continuously improving and optimizing but not necessarily sharing data if the system is nominal. However, once an emergency situation has occurred, emergency information can be sent to the appropriate emergency response services in the Smart City Core network, local cameras and road sensors. Extended information can also be shared with relaxed latency demands to support model updates/training.
* Vehicle sensing for vehicle dynamics under ML model sensing and control. Local ML models sense and react to skids, slides, acceleration and breaking. These models are continuously improving and optimizing but not necessarily sharing data if the system is nominal. However, once an emergency situation has occurred, emergency information is immediately broadcasted to appropriate emergency response services and other system applying related ML models to inform them. Extended information can also be shared with relaxed latency demands to support improve the systems prediction accuracy.

Traffic Surveillance Camera to Metro Traffic Control

Camera senses accident using local AI/ML models and sends notification to traffic control. Traffic control may use multiple cameras to cover the same incident and, thereby, improve its inference accuracy. Extended information exchanged with low-latency demands can be used to re-train and redistribute reference AI model layers accordingly.

Examples of AI-ML Model data exchange are:

* Camera and Road sensors detect emergency situations – Local camera and road sensor ML models are constantly improving their prediction of emergency situations. Once an emergency situation has occurred emergency information data is shared with appropriate systems to improve the response for all emergency responders. Extended information can also be shared with relaxed latency demands to support improve the systems prediction accuracy.

Metro Traffic Control to Metro Emergency Services Network

AI/ML models use automation and prediction to send appropriate EMS vehicles to the accident site.

Examples of AI-ML Model data exchange are:

* Camera, Road and Weather sensor use ML models. ML models sensors are continuously improving and optimizing prediction results. Once an emergency situation is detected, emergency information from these systems can be shared with emergency vehicles to, e.g. improve their routing time through traffic based on the situation.

Stalled Vehicle to Manufacturer

Stalled vehicle sends data (possibly pre-processed, e.g. AI/ML model data) to automobile manufacturer to help diagnose the problem. Manufacturer uses the information to improve product quality, reliability and performance. Federated and/or Distributed Learning techniques can be used to improve on-board vehicle AI/ML model.

Stalled Vehicle to Metro Traffic Control (MTC)

Stalled vehicle sends emergency information to MTC to help warn others. E.g., The specific blind curve location along with other regional data is used to improve all MTC responses. Extended information can also be shared with relaxed latency demands to improve MTC response models.

Stalled Vehicle to Repair Service

Stalled vehicle sends data (possibly pre-processed, e.g. AI/ML model data) to repair service to ensure appropriate response vehicles provide proper tools and equipment. Repair service response use Federated and/or Distributed Learning techniques for constant model improvement.

Approaching Vehicle communication with Metro Traffic Control

MTC warns approaching vehicle of upcoming danger and depending on the SAE level of driving automation, the vehicle responds appropriately. Extended information can also be shared with relaxed latency demands to support improve the systems prediction accuracy.

### 6.5.4 Post-conditions

All systems identified in this scenario use independent, networked AI/ML Distributed or Federated Learning algorithms to aggregate information from multiple sources to improve layer of the system AI/ML model. Independent AI/ML training systems ensure that improved models are distributed back to each end-system to improve the overall safety and robustness of the next response.

### 6.5.5 Existing features partly or fully covering the use case functionality

None.

### 6.5.6 Potential New Requirements needed to support the use case

Table 6.5.6-1 provides example ML models (sizes and DL data rates) that are considered for the use-cases above and Table 6.5.6-2 outlines latency requirements. It is assumed that a maximum of 10% of the (full) model size is exchanged among the participating systems.

Table 6.5.6-1: ML Networks/models and potential DL data rates

|  |  |  |  |
| --- | --- | --- | --- |
| **DNN model** | **32 bits per parameter** | | |
| **Full Model (MBytes)** | **Exchanged data (Mbytes) [10% of full model]** | **Max DL user data rate (Mbit/s)** |
| 1.0 MobileNet-224 [54] | 16.8 | 1.68 | 13.4 |
| SSD-ResNet34 [55] | 81 | 8.1 | 64.8 |
| SSD-MobileNet-v1 [56] | 27.3 | 2.37 | 21.8 |
| MASK R-CNN [57] | 245 | 24.5 | ~100 |
| DLRM [58] | 400 | 40.0 | 10 |

Table 6.5.6-2: Potential Latency Requirements

| **User application** | **Potential Latency Requirements** |
| --- | --- |
| **Exchanged data – Download latency** |
| Vehicle Detects Fault and Pulls Over to Avoid Accident | ~500ms – 1sec |
| Roadside Camera Detects Road Hazard and Warns Smart City Network | ~500ms – 1sec |
| On-Coming Traffic Detects Hazard and Avoids Accident | ~500ms – 1sec |
| Video recognition | ~500ms – 1sec |
| Smart City Detect Accident and Issues Local Warning | ~up to few sec |
| Car Manufacturer/Insurance warned about fault | Up to few mins |

Requirements:

[P.R.6.5-001] The 5G system shall be able to support downloading of data with a maximum size of ~2-40 MB to update the local AI/ML model with latency up to 500ms – 1s.

[P.R.6.5-002] The 5G system shall be able to support downloading of data with a maximum size of ~2-40 MB to update the local AI/ML model with (user experienced) DL data rate of up to 100 Mbit/s.

[P.R.6.5-003] The 5G system shall be able support downloading of data to update the local AI/ML model with communication service availability up to 99.999 %.

## 6.6 Shared AI/ML model monitoring

### 6.6.1 Description

AI/ML models are trained on a training data set to accomplish an established task. The tasks may vary from image or speech recognition to forecasting to optimize, as example, handover performance (see 3GPP TR 28.809 [59]) or tuning Core Network assisted parameters (see clause 5.4.6.2, TS 23.501).

In each of these tasks, the provider of the shared AI/ML model may benefit from sharing a trained AI/ML model with the consumer(s) of the shared AI/ML model or may benefit from a distributed/federated AI/ML model training or a split AI/ML model training over the 5G system. Furthermore, AI/ML model monitoring is a requirement to enable online learning in the network (e.g. via Reinforcement Learning), a set of techniques more suited to promptly react to service degradation.

NOTE: The following terms are used in this use case:

**Shared AI/ML model**: AI/ML model that is shared among different applications, e.g., the AI/ML model is pre-trained and provisioned to different consumers, or the AI/ML model is trained using distributed/federated learning approach or by splitting the model training phase to different parts executed in different network locations.

**Shared AI/ML model provider**: application server that is providing or managing a "shared AI/ML model".

**Shared AI/ML model consumer**: application, e.g. running on the UE, that is using/consuming a "shared AI/ML model".

Due to changes in the scenario (i.e., in the context from which training data are collected), an AI/ML model may provide poor performances compared with the performance of the AI/ML model measured during the model testing phase. This can happen when over time the distribution of the input data for inference differs from the distribution of the training data, or if the AI/ML model is utilized in a different context. In this case, the shared AI/ML model provider should be able to promptly detect the performance degradation and react in order to avoid service degradation or disruptions. Quite often, update of a shared AI/ML model is not solely dependent on inference results of one shared AI/ML model consumer as performance degradation could be due to some other error source, e.g., model input measurement errors. To detect outdated shared AI/ML model, the shared AI/ML model provider can make use of inference results from multiple shared AI/ML model consumers and a spatial and temporal analysis is performed before triggering shared AI/ML model update.

Therefore, the shared AI/ML model provider, once sharing an AI/ML model over 5G System with a shared AI/ML model consumer, needs to keep track of the model performances to detect possible performance degradation of the shared AI/ML model (e.g. based on inference feedback from AI/ML model consumer such as a lower confidence level).

Alternatively, the shared AI/ ML model provider can split the AI/ ML model training with a shared AI/ML model consumer to constantly improve the performance of the shared AI/ ML model based on a local training and/or inference feedback from the shared AI/ML model consumer. The local part of the shared AI/ML model will be trained/fine-tuned under the shared AI/ML model provider guidance. The input to this training will be data available at the shared AI/ML model consumer. The output of local model training or an inference at shared AI/ML model consumer can be provided to the shared AI/ML model provider and be used by the shared AI/ML model provider to provide further information for the 5G System to improve its operations.

### 6.6.2 Pre-conditions

The shared AI/ML model provider stores multiple AI/ML models along with their performances measured during the test phase.

The shared AI/ML model provider is capable of sharing AI/ML models with shared AI/ML model consumers leveraging on 5GS.

The AI/ ML model provider is capable of splitting and/or distributing the AI/ML model training with/to a shared AI/ ML model consumer leveraging on 5GS.

The shared AI/ML model consumer may run applications requiring the usage of AI/ML models and download them from the AI/ML model provider via the 5GS.

### 6.6.3 Service Flows

* 1. The shared AI/ML model provider wants to optimize performance of some process by means of shared AI/ML model performance.
  2. The shared AI/ML model provider sends the trained shared AI/ML model to the shared AI/ML model consumer at the UE leveraging on the 5GS.
  3. The UE receives the model and employs the model to perform local training and inference using data available on the UE.
  4. The shared AI/ML model provider monitors the context scenario (e.g. the UE data which is available to the application) in which the UE is running the shared AI/ML model and the model performance.
  5. If a change in the context scenario or model performance is detected, e.g. based on an inference feedback from the shared AI/ML model consumer, the shared AI/ML model provider, in order to avoid model performance degradation, shares with the AI/ML model consumer an updated version of the shared AI/ML model retrained to capture the new context with expected better performance.
  6. The UE continues to run the updated model without experiencing performance degradation.

### 6.6.4 Post-conditions

Following the example in the use case, the shared AI/ML model provider receives an accurate forecast regarding AI/ML model performance, and the AI/ML model consumer is using an AI/ML model with high performance.

### 6.6.5 Existing features partly or fully covering the use case functionality

### 6.6.6 Potential New Requirements needed to support the use case

[P.R.6.6-001] The 5GS shall be able to transfer an updated AI/ML model from the shared AI/ML model provider to the shared AI/ML model consumer within [1s-1min] latency for AI/ML models of a maximal size of [100-500] MB.

## 6.7 Prediction of AI/ML model distribution

### 6.7.1 Description

If the AI/ML model server was intelligent enough, it would be ideal that the AI/ML model server could predict that the User would need to use some AI/ML model and download the AI/ML model to the UE in advance.

Take image recognition for example, the AI/ML model could be able to predict and download specific AI/ML model in advance based on the user’s location, direction of travel and users use of image recognition functionality. For the users located in a famous history museum or moving towards a famous history museum, the 5G network functions and the AI/ML model server could interact with each other and predict that the users could possibly need to use the AI/ML model designed specifically for such history museum which is suitable for the indoor environment and good at historic pictures recognition. Similarly, for the users located in a zoo or moving towards a zoo, the 5G network functions and the AI/ML model server could interact with each other and predict that the users could possibly need to use the AI/ML model designed specifically for animal recognition. In contrast, for the users located in an arboretum or moving towards an arboretum, the 5G network functions and the AI/ML model server could interact with each other and predict that the users could possibly need to use the AI/ML model designed specifically for flower or plant recognition.

Editor's Note: it is FFS whether other examples than image recognition need to be studied.

The 5G network and the AI/ML model shall not violate the user privacy policies for the prediction or pre-download of AI/ML models.

According to user privacy policies, the AI/ML model server may not to be able to get access to the UE’s location information.

If the AI/ML model server is allowed to access to the UE’s location information according to user privacy policies, the AI/ML model server could obtain the UE’s location information via two options:

Option 1: the AI/ML model server could obtain the UE’s location information directly from the UE, over the application layer. This is pure application layer interaction between the AI/ML based image recognition application and the AI/ML model server.

Option 2: the AI/ML model server could obtain the UE’s location information via the location service provided by the 5G network. This is already specified in TS 23.273 [6] (see further analysis in section 6.1.4).

If the AI/ML model server is not allowed to access to the UE’s location information according to user privacy policies, the 5G network has the UE’s location information and is not allowed to expose the UE location information to the AI/ML model server. The 5G network function could still interaction with the AI/ML model server for prediction of need of certain AI/ML model without breaking the user privacy policies. There could be lots of workable solutions. Here daftly describe two possible solutions:

Option a): Prediction based on monitoring function of the 5G network. New monitoring event could to be introduced to assist the prediction by the AI/ML model server without disclosure of User location information to the AI/ML model server. Option b) Prediction based on NWDAF analytics function of the 5G network. Based on analytics of UE’s location, mobility, download data size and etc, the 5G network could predict that lots UE will download certain amount of data from an AI/ML model server in some location area and inform the AI/ML model that certain UE will probably download certain amount of data. The AI/ML model server could use such information to adjust its prediction.

### 6.7.2 Pre-conditions

The UE runs an application providing the capability of AI/ML model inference for image recognition.

An AI/ML server manages the AI/ML model pool, and is capable to download the requested model to the application providing AI/ML based image recognition.

The 5G system is able to provide 5G network related information to the AI/ML server.

### 6.7.3a Service Flows for Prediction based on monitoring function of the 5G network

1) The AI/ML model server provides to the 5G network function with some criteria. Each criterion could have different rules. For example, one criterion could contain a list of location areas which contains all the museums, galleries and antique shops all over the country.

2) If the 5G network functions determine the UE meets one criterion, for example, the UE is located in the list of location areas, then the 5GS network function (e.g., NEF) notify the AI/ML model server which criterion the UE meets.

3) According the criterion that the UE meets, the AI/ML model server determines that the UE might need to use the AI/ML model designed specifically for such history museum which is suitable for the indoor environment and good at historic pictures recognition.

4) The 5G operator need to make sure that the criteria provided by the AI/ML model does not break the user privacy policies.

### 6.7.3b Service Flows for Prediction based on NWDAF analytics function of the 5G network

1) The NWDAF do the data collection and analytics work for the UE’s location, mobility, download data size etc.

2) Based on the NWDAF analytics, the 5G network could predict that lots UE will download certain amount of data from an AI/ML model server in some location area.

3) For the AI/ML model server that has subscribed the data analytics service from 5G network, 5G network inform AI/ML model server directly or via NEF that certain UE will probably download certain amount of data.

4) The AI/ML model server could use such information to adjust its prediction.

### 6.7.4 Post-conditions

The 5G network and the AI/ML model server could predict the need of download certain AI/ML model from the UE and refine the prediction during the interaction.

The 5G network could adjust its resource for download of data from AI/ML without expansion of capacity based on the prediction.

The AI/ML model server could predict the user preference and needs without knowing UE’s private information, such as location information.

The UE could get better user experience during the use of AI/ML model application via 5G network.

### 6.7.5 Existing features partly or fully covering the use case functionality

TS23.273[6] clause 4.1:

Location information for one or multiple target UEs may be requested by and reported to an LCS client or an AF within or external to a PLMN, or a control plane NF within a PLMN. Location information contained in the location request and location information contained in the location response are defined in clause 5.5.

For location request from LCS client (neither in the UE nor in the NG-RAN) or AF external to a PLMN, privacy verification of the target UE shall be enabled to check whether it is allowed to acquire the UE location information based on UE LCS privacy profile and whether the LCS client or the AF is authorised to use the location service as defined in clause 5.4.

Note: the AI/ML server plays the role of AF to acquire the UE location information.

TS 23.501[48] clause4.15.3.1

Currently the Monitoring Events feature defined in TS 23.501 [48] is intended for monitoring of specific events in the 3GPP system and reporting such Monitoring Events via the NEF. Location Reporting is one of the supported monitoring events is specified in Table 4.15.3.1-1 of TS 23.502 [49]. However, the current monitoring event features indicates either the Current Location or the Last Known Location of a UE to an AF via NEF, this is not applicable when the AI/ML model server is not allowed to access to the UE’s location information according to user privacy policies.

### 6.7.6 Potential Requirements

[P.R.6.7-001] Subject to user consent, operator policy and regulatory constraints, the 5G system shall support the provision of monitoring information or analytics information to a trusted 3rd party AI/ML server for allowing this 3rd party AI/ML server to make a prediction for a suitable AI/ML model to be downloaded to the concerned UE.

# 7 Distributed/Federated Learning over 5G system

## 7.1 Uncompressed Federated Learning for image recognition

### 7.1.1 Description

Nowadays, the smartphone camera has become the most popular tool to shoot image and video, which holds many valuable vision data for image recognition model training. For many image recognition tasks, the images/videos collected by mobile devices are essential for training a global model. Federated Learning (FL) is an increasingly widely-used approach for training computer vision and image recognition models.

In Federated Learning mode, the cloud server trains a global model by aggregating local models partially-trained by each end devices based on the iterative model averaging [30]. As depicted in Figure 7.1-2, within each training iteration, a device performs the training based on the model downloaded from the AI server using the local training data. Then the device reports the interim training results (e.g., gradients for the DNN) to the cloud server via 5G UL channels. The server aggregates the gradients from the devices, and updates the global model. Next, the updated global model is distributed to the devices via 5G DL channels, the devices can perform the training for the next iteration.

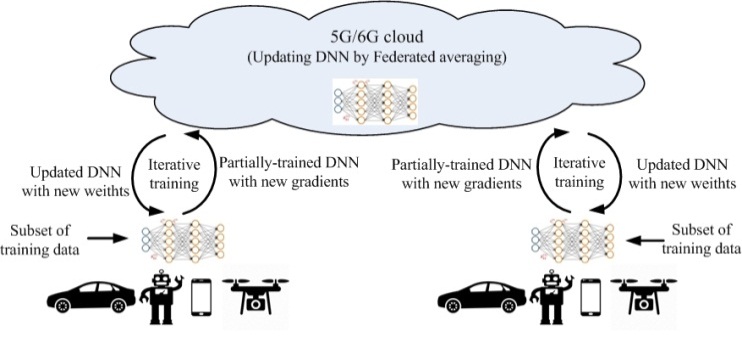


Figure 7.1.1-1. Federated Learning over 5G system

An iterative Federated Learning procedure is illustrated in Figure 7.1.1-2. In the *N*th training iteration, the device performs the training based on the model downloaded from the FL training server using the images/videos collected locally. Then the device reports the *N*th-iteration interim training results (e.g., gradients for the DNN) to the server via 5G UL channels. Meanwhile, the global model and training configuration for the (*N*+1)th iteration are sent to the device. When the server aggregates the gradients from the devices for the *N*th iteration, the device performs the training for the (*N*+1)th iteration. The federated aggregation outputs are used to update the global model, which will be distributed to devices, together with the updated training configuration.

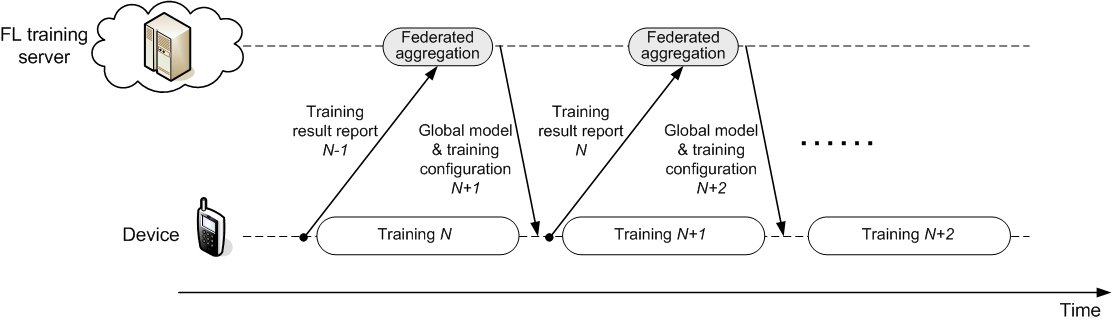


Figure 7.1.1-2. Federated Learning timeline for image recognition

In order to fully utilizing the training resource at device and minimizing the training latency, the training pipeline shown in Figure 7.1.1-2 requires the training results report for the (N-1)th iteration and the global model/training configuration distribution for the (N+1)th iteration are finished during the device’s training process for the Nth iteration. The analysis in Section 7.1.6 will be developed based on the processing timeline. In practice, more relaxing FL timeline can also be considered with sacrificing the training convergence speed.

The training time should be minimized since mobile devices may only stay in an environment for a short period of time. Further, considering the limited storage at device, it may not realistic to require the training device to store a large amount of training data in the memory for a training after it moves outside the environment.

Different from the decentralized training operated in cloud datacenters, Federated Learning over wireless communications systems need to be modified to adapt to the variable wireless channel conditions, unstable training resource on mobile devices and the device heterogeneity [10, 32, 34]. The Federated Learning protocol for wireless communications can be depicted in Figure 7.1.1-3 [10, 31-32].

For each iteration, the training devices can firstly be selected. The candidate training devices report their computation resource available for the training task to the FL server. The FL server makes the training device selection based on the reports from the devices and other conditions, e.g. the devices’ wireless channel conditions, geographic location, etc..

Hereby, besides performing federated learning task, the training devices in a communication system have their other data to transmit at uplink (e.g. for ongoing service transactions), that may be high priority and not latency-tolerant and its transmission may affect a device's ability to upload the locally trained model. Device selection must therefore account for a trade-off to upload the training results as compared to uploading other uplink data. Furthermore, skipping a device from federated learning model aggregation for one or more iterations affects the convergence of the federated learning model. Therefore, candidate training device selection over wireless links is more complex as compared to federated learning in data centers.

After the training devices are selected, the FL server will send the training configurations to the selected training devices, together with global model for training. A training device starts training based on the received global model and training configuration. When finishing the local training, a device reports its interim training results (e.g., gradients for the DNN) to the FL server. In Figure 7.1.1-3, the training device selection is performed, and the training configurations are sent to the training devices at the beginning of each iteration. If the conditions (e.g. device’s computation resource, wireless channel condition, other service transactions of the training devices) are not changed, the training device re-selection and training re-configuration might not be needed for each iteration, i.e. the same group of training devices can participate the training with the same configuration for multiple iterations. Still, the selection of training devices should be alternated over time in order to achieve an independent and identically distributed sampling from all devices, i.e., give a fair chance to all devices to contribute to the aggregated model.

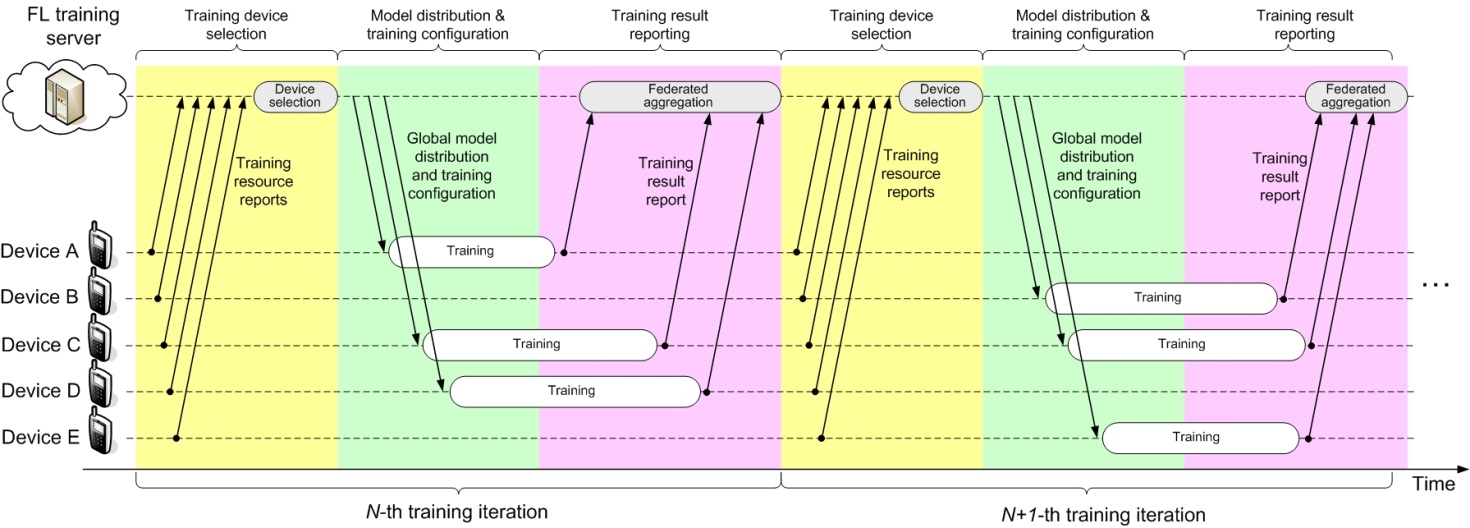


Figure 7.1.1-3. Typical Federated Learning protocol over wireless communication systems

### 7.1.2 Pre-conditions

The UE runs an application providing the capability of Federated Learning (FL) for the image recognition task.

The FL application on the UE is capable to report its interim training results to the FL server.

The FL server is capable to aggregate the interim training results from the federated UE, form the global model, and distribute the global model for training in the next iteration.

The 5G system has the ability to provide 5G network related information to the FL server.

### 7.1.3 Service Flows

1. The FL server selects a set of federated UEs based on different own criteria. One possibility is to use available information on UE geographic location (subject to user consent and/or operator/regulatory requirements).
2. The FL server distributes the global model to be federated UEs via 5G network.
3. The FL application in a federated UE performs the training based on the local training data set collected by the UE, and then reports the interim training results (e.g., gradients for the DNN) to the FL server via 5G network.
4. The FL server aggregates the gradients from the UEs, and updates the global model.
5. Redo Step 1) to 4) for the training for the next iteration.

### 7.1.4 Post-conditions

The AI/ML model for image recognition is trained and converges, and the training accuracy and latency need to be guaranteed.

The FL training task for image recognition can be completed under the available computation and energy resource of the federated UEs. And the consumed the computation, communication and energy resources over the federated UEs and the FL server are optimized.

### 7.1.5 Existing features partly or fully covering the use case functionality

This use case mainly requires high data rate together with low latency. The high data rate requirements to 5G system are listed in Clause 7.1 and 7.6 of TS22.261 [4]. As in Table 7.1-1 of [4], 300Mbps DL experienced data rate and 50Mbps UL experienced data rate are required in dense urban scenario, and 1Gbps DL experienced data rate and 500Mbps UL experienced data rate are required in indoor hotspot scenario. As in Table 7.6.1-1 of [4], cloud/edge/split rendering-related data transmission requires up to 0.1Gbps data rate with [5-10]ms latency countrywide.

### 7.1.6 Potential New Requirements needed to support the use case

As introduced in clause 7.1.1, in order to minimizing the training latency for Federated Learning for image recognition, the computation resource at device for the training task should be fully utilized, i.e. the training pipeline in figure 7.1.1-1 is desired to be maintained.

If considering to train a 7-bit CNN model VGG16\_BN using 224×224×3 images as training data, table 7.1.6-1 shows the sum of gradient uploading latency, the federated aggregation latency and the global model downloading latency should be no larger than the GPU computation time at device for one iteration. For different batch sizes, the gradient uploading and the global model downloading for each iteration needs to be finished in [52~162ms], respectively.

Different from the “single-UE latency” considered by previous requirement study [15], what is more essential for synchronous Federated Learning is the latency within which all federated devices can finish the gradient uploading. In other words, all training devices need to finish the gradient uploading within the latency in table 7.1.6-1, even if multiple training devices are present in a cell.

The size of the 8-bit VGG16\_BN model is 132MByte for either the trained gradients or the global model. Hence in order to finish the gradient uploading and the global model downloading within the duration, the required UL and DL data rate are shown in table 7.1.6.1-1, which are [6.5Gbit/s to 20.3Gbit/s] respectively. And it should be noted that 132MByte is the size without compression. The size may be reduced if advanced model/gradient compression techniques can be adopted.

In the legacy requirements to 5G system, e.g. [4], the full coverage is always desired for all UEs. However, the AI/ML model training task may to some extent relax the requirements on continuous network coverage. When a FL server selects the training devices for a Federated Learning task, it can try to pick the UEs in a satisfactory coverage, if they can collect the training data needed. This implies that even in a non-continuous coverage of 5G mmWave, the Federated Learning task can be well carried out. This provides to 5G operators a service better exploring the use of their FR2 spectrum resource.

#### 7.1.6.1 Potential KPI Requirements

The potential KPI requirements needed to support the use case include:

[P.R.7.1-001] The 5G system shall support a user experienced DL data rate as given in table 7.1.6.1-1, to enable uncompressed Federated Learning for image recognition.

[P.R.7.1-002] The 5G system shall support a user experience UL data rate as given in from table 7.1.6.1-1, to enable uncompressed Federated Learning for image recognition.

Table 7.1.6.1-1: Latency and user experienced UL/DL data rates for uncompressed Federated Learning

| **Mini-batch size**  **(images)** | **GPU computation time (ms)** | **Maximum latency for trained gradient uploading and global model distribution (see note 1)** | **User experienced UL/DL data rate for trained gradient uploading and global model distribution (see note 2)** |
| --- | --- | --- | --- |
| 64 | 325 | 3.24s | 325Mbit/s |
| 32 | 191 | 1.9s | 55Mbit/s |
| 16 | 131 | 1.3s | 810Mbit/s |
| 8 | 111 | 1.1s | 960Mbit/s |
| 4 | 105 | 1.04s | 1.0Gbit/s |
| NOTE 1: Latency in this table is assumed 20 times the device GPU computation time for the given mini-batch size.  NOTE 2: Values provided in the table are calculative needs for an 8-bit VGG16 BN model with 132MByte size, given mini-batch sizes and a duration of [FFS] seconds per iteration. Necessary user experience UL/DL data rates can be reduced by e.g. setting longer times per iteration, applying compressed FL, or using another AI/ML model. | | | |

## 7.2 Compressed Federated Learning for image/video processing

### 7.2.1 Description

Federated learning can be used to train AI/ML models based on number of images and videos generated by cameras in mobiles devices by iteratively exchanging gradient of updating models instead of direct user images and videos. Because this method can utilize images and videos from many users, the performance of a trained AI/ML model can be significantly higher than a stand-alone case. However, the basic federated learning methods can have disadvantages by massive uplink traffics and high computational cost at a mobile device. Therefore, it is beneficial to consider a compressed federated learning (CFL) method, which allows compressed (not full) models to be transferred during a learning period.

Figure 7.2.1-1 shows the essential procedure of CFL. CFL iteratively performs a set of the three operation stages. In order to describe the iterations in CFL, we introduce a cycle of I: each cycle begins with the 1st iteration and ends with I-th iteration, which is immediately followed by the 1st iteration of the next cycle (e.g., (I+1)st iteration). For each iteration, the three operation stages include the training UE selection, the sparse weight distribution, and the training result reporting stages. The operations of these three stages in the first iteration and the operation in the last iteration are different from the operations in the other iterations.

Each iteration in CFL starts with the training UE selection stage, at which the CFL server selects a set of available users from the candidate users to associate with the same purpose AI/ML model. To the selected users, ready to participate in the learning process because of being an available state, the CFL server transmits the train configuration information. At the next stage, the CFL server sends the sparse global model, which could be an initial version of the AI/ML model in the first iteration. Otherwise, the sparse global model is an aggregated version based on user reporting information.

Then, each UE trains a received model after expanding the spatial model and reports an intermediate training result to the CFL server, where the training result is comprised only of significant value weight gradients for applying a model compression. By doing so, uplink throughout requirement can be significantly reduced in comparison with the basic federated learning method without compression. In the last iteration, I-th iteration, the CFL Server sends ‘train stop message’ to UEs so that the UEs can stop sending its update any longer, and the CFL Server performs fine-tuning by pruning unnecessary nodes. Throughout these multiple iterations of a cycle (i.e., from the 1st iteration through the I-th iteration) as in the figure, the AI/ML model will be progressively enhanced based on user data in mobile networks at reduced requirements of uplink and downlink throughput.

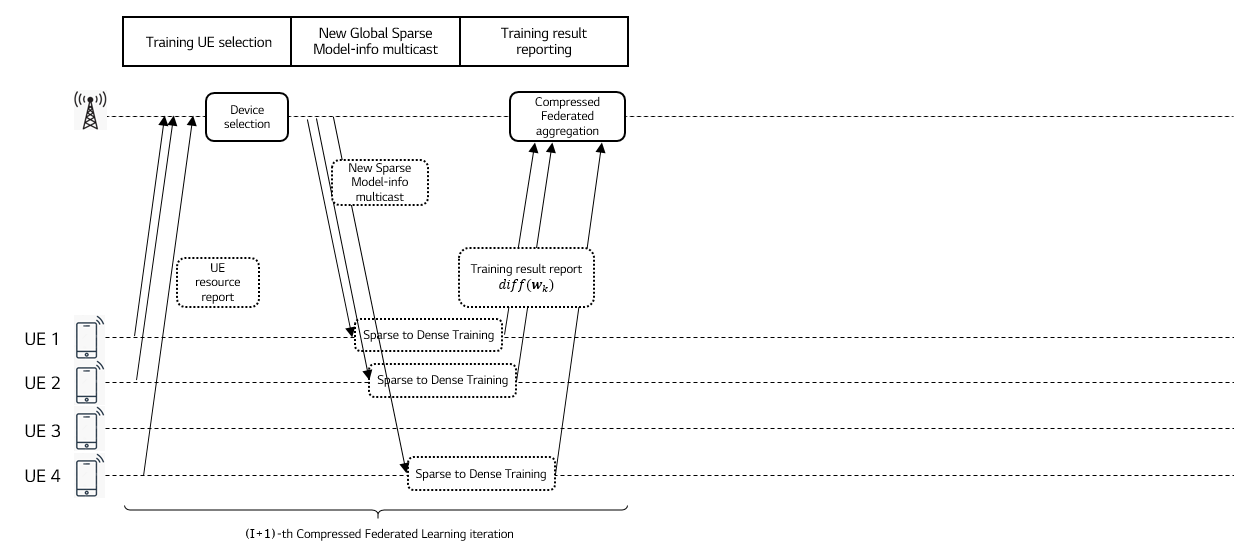
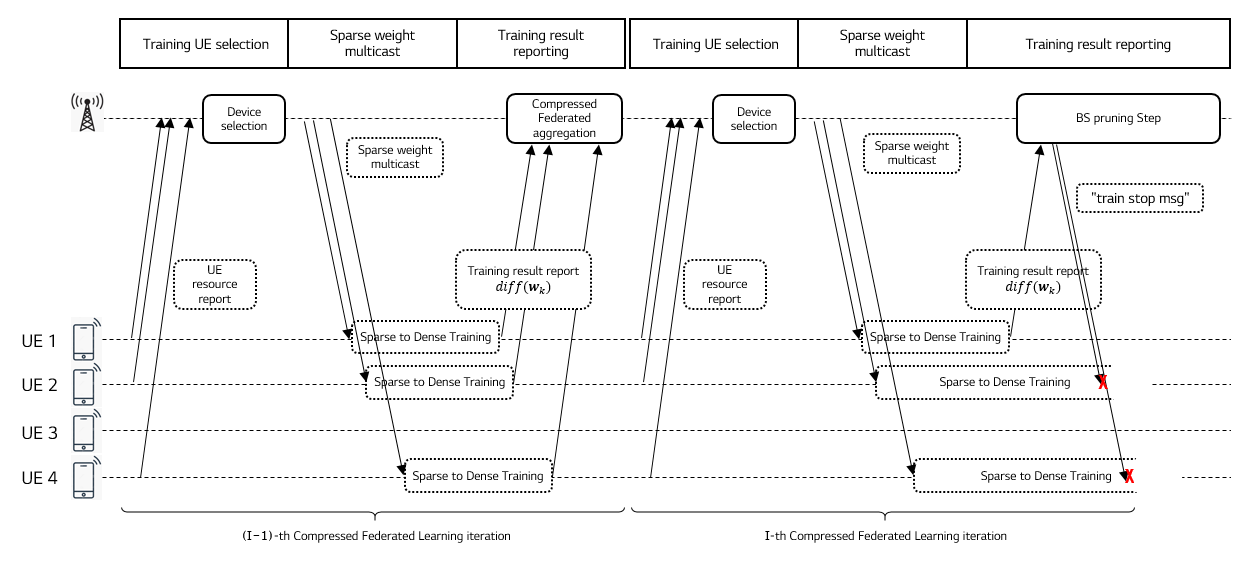
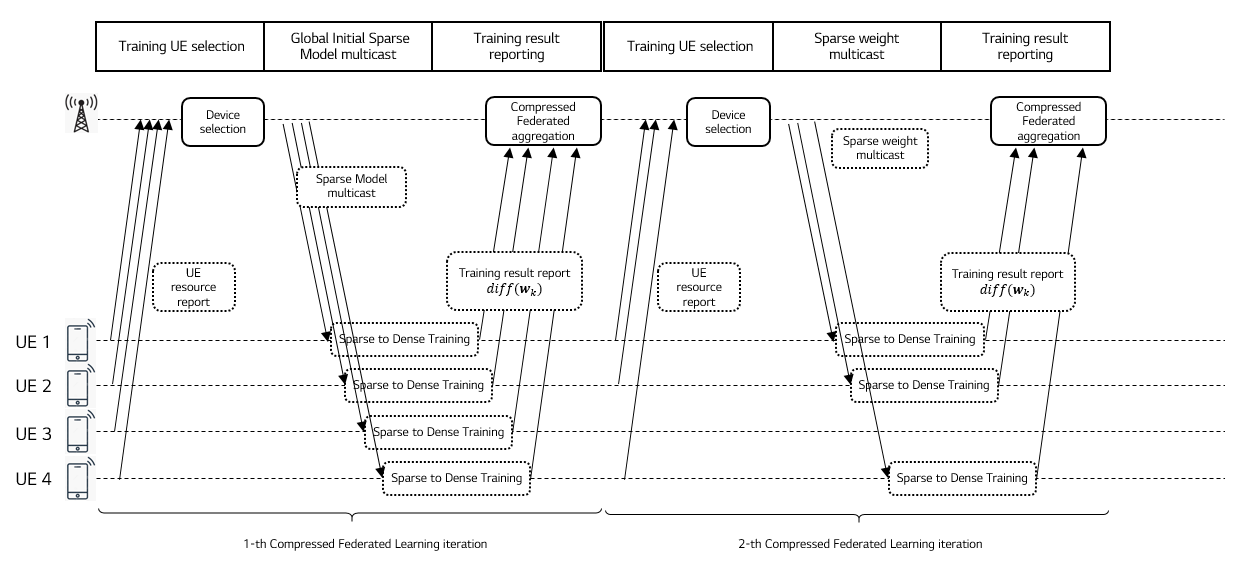


Figure 7.2.1-1. Compressed Federated Learning timeline for image recognition

### 7.2.2 Pre-conditions

UE can have a computational hardware and algorithm capability to train an AI/ML model such as for an image and video cognition.

UE can send intermediate training results to a CFL server.

A CFL server can select training devices and determine training configuration.

A CFL server can aggregate intermediate training results and generate a sparse global model for the next learning iteration.

A CFL server can distribute a global AI/ML mode to a set of selected users.

### 7.2.3 Service Flows

Step 1: The CFL server selects the training users from candidate users.

Step 2: The CFL server sends the configuration information to the selected users.

Step 3: The CFL server distributes the initial (or, aggregated) sparse global model to the selected users through a 5G networks.

Step 4: Each UE expands the sparse global model and train the expanded model using its local data. Then, each UE sends only significant value weight gradients to the CFL server.

Step 5: The CFL server aggregates the training results received from the training UEs and update a global model using the aggregated results.

Step 6: Until the AI/ML model reaches saturated performance enhancement, the process runs repeatedly from step 1.

Step 7: Otherwise, the CFL server performs fine-tuning for a global model compression for a global model. This process can be applied regularly so as to improve bandwidth and computation resource efficiency before the training finalization.

Finally, the CFL server distributes the new sparse global model to all users which needs the same AI/ML model.

### 7.2.4 Post-conditions

For a UE prospective, CFL can reduce uplink and downlink throughput requirements for the federated learning process. Also, the computational complexity in UEs can be significantly reduced because of enabling a compressed model.

### 7.2.5 Existing features partly or fully covering the use case functionality

**Latency analysis for gradient uploading and the global model downloading for image recognition**

AI/ML model training data for CFL is a new type of traffic. Consider CFL to train an 8-bit CNN VGG16 model with 224x224x3 images. Table 7.2.5-1 shows that the single GPU computation time should be larger than the addition of gradient uploading latency and global model downloading latency.

Table 7.2.5-1: GPU computation time for different mini-batch sizes   
for Compressed Federated Learning

| **Mini-batch size**  **(images)** | **GPU computation time (ms)** | **Required latency for trained gradient uploading (sec)**  **(see note 1)** | **Required latency for global model distribution (sec)**  **(see note 1)** |
| --- | --- | --- | --- |
| 64 | 325 | 3.25 | 3.25 |
| 32 | 191 | 1.91 | 1.91 |
| 16 | 131 | 1.31 | 1.31 |
| 8 | 111 | 1.11 | 1.11 |
| 4 | 105 | 1.05 | 1.05 |
| NOTE 1: Latency in this table is assumed 20 times the device GPU computation time for the given mini-batch size. | | | |

**Data rate analysis for gradient uploading and the global model downloading for image recognition**

Table 7.2.5-2 shows the required data rate for gradient uploading and global model downloading for the above 8-bit VGG16 model when CFL is applied. We calculated the required data rate based on table 1 in [39], in which the pruning size of the 8-bit VGG16 model can be reduced 13 times from the original size of 138 Mbyte. It is noteworthy that 13 times model compression gives almost no accuracy degradation for the 8-bit VGG16 model. If we assume that the minibatch size is 4, the uplink required rate is compressed trained parameter size \* 8 / (GPU computation time / 2) = (138 / 13) Mbyte \* 8 bits / ((105ms\*20/1000) / 2) = 80.88 Mbit/s which is same to the downlink require rate.

Table 7.2.5-2: User experienced data rate for gradient uploading and global model downloading  
User experienced data rate = compressed trained parameter size \* 8 / (GPU computation time / 2)

| **Mini-batch size**  **(images)** | **User experienced UL data rate for trained gradient uploading (Mbit/s)**  **(see note 2)** | **User experienced DL data rate for global model distribution (Mbit/s)**  **(see note 2)** |
| --- | --- | --- |
| 64 | 26.13 | 26.13 |
| 32 | 44.46 | 44.46 |
| 16 | 64.83 | 64.83 |
| 8 | 76.51 | 76.51 |
| 4 | 80.88 | 80.88 |
| NOTE 2: Values provided in the table are calculative needs for an 8-bit VGG16 BN compressed model with 10.61MByte size, given mini-batch sizes per iteration. | | |

In the case of 8-bit CNN VGG16, CFL compared to FL can transmit up to 13 times more model information through model compression. Hence, when using the same uplink payload as FL, up to 13 times more users can be simultaneously supported. For example, as shown in Figure 7.2.5-1 which is redrawing using the associated data in [50], the accuracy performance of the FL with 15 participants, which is five times than the FL with 3 participants, can be significantly increased by 6.5%. If we consider the maximum compression efficiency of CFL with supporting 13 times more users, a further performance gain is expected.

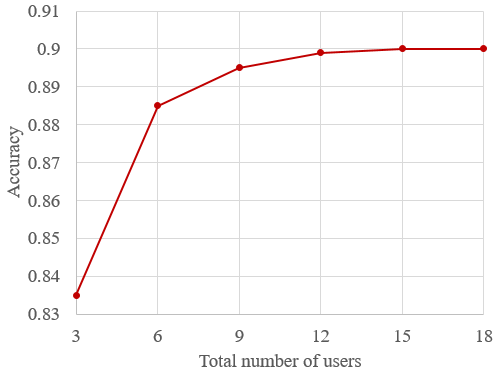


Figure 7.2.5-1: Federated Learning accuracy as the total number of users   
Redrawing using the associated data in [60]

For the flexibility and accuracy of FL, the compression rate of CFL can be adjusted according to the user's channel conditions. For example, even in the case of a user with a bad channel, the problem of not participating in FL can be solved by increasing the compression ratio of the model.

### 7.2.6 Potential New Requirements needed to support the use case

[P.R.7.2.6-001] The 5G system shall support to upload a trained gradient for each iteration of Compressed Federated Learning with a maximum latency of 1.05~3.25s.

[P.R.7.2.6-002] The 5G system shall support downloading the global model for each iteration of Compressed Federated Learning with a maximum latency of 1.05~3.25s.

[P.R.7.2.6-003] The 5G system shall support UL unicast transmission with 26.13~80.88Mbit/s user experienced UL data rates and a communication service availability not lower than [99.9%] for reporting the trained gradients for Compressed Federated Learning.

[P.R.7.2.6-004] The 5G system shall support DL multicast transmission with 26.13~80.88Mbit/s user experienced DL data rates and a communication service availability not lower than [99.9%] for distributing the global model for Compressed Federated Learning.

## 7.3 Data Transfer Disturbance in Multi-agent multi-device ML Operations

### 7.3.1 Description

A brief story of machine learning is nothing but a computer (that has no or limited imprinted programs for a certain task) exploiting its own capability (“performance”) towards a certain task using data (“experience”). There are several criteria to classify the types of Machine Learning depending on the characteristics of the method used. This use case is intended to describe a case of multi-agent multi-device ML operations with heavy data (i.e., the data size is huge) when there is partial or total disturbance for data collection/transfer (e.g., privacy regulation or temporary technical limitation like shortage of network resources or temporary failure). As depicted in Fig. 7.3.1-1, this use case, Part I, is specifically related to a scenario that there are multiple agents and multiple collecting devices where the devices can perform ML operations, not necessarily in full but as much as they can (i.e., functional splitting is possible between a device and one or more learning agents).

NOTE 1: MBL (Multiple Batch Learning) is one of examples that belong to this scenario, which is known to be better performing when the data is too big for a single agent/server to handle.

NOTE 2: A learning agent is a type of AI/ML training servers that can be inside 5GS (e.g., AI/ML-based support for communication) or outside 5GS (e.g., communication support for AI/ML). Multi-agent training has multiple collaborating agents involved as depicted in Fig. 7.3.1-1 where dotted lines denote collaboration relation between two or more learning agents. The respective learning agents may have different operational situations (e.g., different technical issues, such as congestion or breakdown, and different jurisdiction-related restrictions). In this use case, it is assumed that the learning agent is typically in the cloud, interacting with UE (as a collecting agent). Learning agents A1, A2, ..., An are not an UE.



Figure 7.3.1-1: Functional relation between multiple devices (M1, M2, …, Mk in the form of UE) and multiple agents (A1, A2, …, An). Data sharing between any pair of agents, if exists, is not disturbed/restricted. In “sharing scenario”, the data would generally be *deep-processed data* as opposed to *simply pre-processed* or *raw data*.

In the age that privacy was not affecting the flow of data from the source to the learning agent (e.g., a computer), the expected performance is the outcome of all possible computational considerations of the data collected (e.g., refer to the green solid curve of Fig. 7.3.1-2). However, if there is a certain level of disturbance in data collection, the achievable performance toward the given task would not be as good as the one with no such disturbances (refer to the gap between the green solid curve and blue dotted curve in the same figure). Some examples of such disturbances include:

1. privacy regulations, such as EU’s General Data Protection Regulation (GDPR) or California Consumer Privacy Act (CCPA);
2. limited capability of transport-layers, such as lack of network resources (e.g., radio resources due to temporal degradation, higher noise/interference level, highly crowed situations, partial/total break-down, and so on) preventing *input data* from being delivered in time:
   1. case 1: a portion of input data delivered in time, if any, is still useful (i.e., entropy can get increased/ improved)
   2. case 2: a portion of input data delivered in time, if any, is not useful (i.e., it’s not enough to get entropy increased/ improved)

NOTE 3: “input data” can be raw data, trained data, or an intermediate combination of them that is to be transferred from a UE (or a group of UEs, respectively) to one of learning agents.

It is commonly understood that the more data a learning model utilizes the better performance the learning model can achieve (assuming the data are reasonably independent from statistics perspectives or sufficiently correctly labelled (when supervised learning is concerned)), if not too large. However, if there is some disturbance in data collection/transfer, such as regional regulations or technical limitations (as described above), the expected performance would not be as good as the case without the disturbance (the *vertical difference* in figure 7.3.1-2); in addition, it is expected that the learning model would need to take more time to accumulate “experience” with the reduced feeding-rate of learning data caused by such disturbance (the *horizontal difference* in figure 7.3.1-2).



Figure. 7.3.1-2: Performance gap vs. experience of learning for a given task: (1) with disturbance of input data collection/transfer (green, solid) (2) without disturbance of input data collection/transfer (blue, dotted).

Fig. 7.3.1-3 shows an example of possible preparation action that can be taken in UE side if some predictive information can be made available when disturbance is about to happen. In the figure, the preferred deadline for input data transfer is 1 sec (t = t0+1) with the amount of useful “input data” 3 bits in which two different kinds of scheduling are given: (a) is imperfect scheduling whereas (b) is good scheduling as an example, respectively. In reality, the transfer payload type is not limited to “input data” for learning agent(s) and it can also be applied to “learning model” transfer. The transfer direction can be “uplink” (e.g., for input data transfer) or “downlink” (e.g., for model distribution/transfer). This simplified example is intended to explain the justification why new technical requirements would be needed especially when some disturbance exists (e.g., by regulatory or technical causes).



Figure 7.3.1-3: Example of disturbance of input data transfer within a preferred deadline of 1 sec (t = t0+1) with the amount of useful “input data” 3 bits (as an example): (a) For the 3-bit input data amount (“trapezoid” in grey), it takes 2 sec with “imperfect scheduling” (b) For the 3-bit input data amount (“rectangle” in grey), it takes 1 sec with “good scheduling” where the network resources can be assigned to others (other UEs) during (t0+1, t0+2).

Given the regulatory disturbance, it is intended to minimize (or to prepare to minimize) the impact on transfer caused by technology (e.g., scheduling and/or information necessary for 3GPP entity to perform “good scheduling”). The expected requirements and service flow description are as follows.

### 7.3.2 Pre-conditions

There are three UEs M1, M2 and M3 (computers or learning machines in the form of a UE).

There are two learning agents/servers in the cloud.

UEs M1, M2 and M3 collect data and they process the data for learning they have collected or are collecting, if available, but they don’t have to complete the processing due to limited computational capability.

Each UE (M1, M2 and M3, respectively) has a functional splitting point negotiated with their agent(s) regarding data processing for learning.

Agent A1 working with agent A2 for a task can share its outcome with agent A2 so that the outcome from agent A2 can jointly improve the outcome, if possible, which is possibly better than A1’s individual outcome and A2’s individual outcome.

UE M1 and UE M2 are located in Area 1 of some jurisdiction that doesn’t restrict collecting certain type of data.

*Initial connections:*

UE M1 is connected to agent A1 (via eNB, ng-eNB, or gNB) when data connection is necessary (e.g., when needed to upload some data or when needed to download some model).

* Agent A1 provides UE M1 with an alternative agent (i.e., Agent A2) for the use of disturbance, which is (one of) participating agent(s) that agent A1 shares data. [See Description clause for typical types of data]
* UE M1 is transferring learning data to agent 1.

UE M2 is connected to agent A1.

* Agent A1 provides UE M2 with an alternative agent (i.e., Agent A2) for the use of disturbance, which is (one of) participating agent(s) that agent A1 shares data.
* UE M2 is transferring learning data to agent 1.

UE M3 is connected to agent A2.

* Agent A2 provides UE M3 with an alternative agent (i.e., Agent A1) for the use of disturbance, which is (one of) participating agent(s) that agent A2 shares data.
* UE M3 is transferring learning data to agent 2.



Figure 7.3.2-1: Initial connections b/w mobile devices and agents

### 7.3.3 Service Flows

While transferring learning data to agent A1, UEs M1 and M2 move into Area 2 of some jurisdiction that has restrictive regulations for agent A1 to collect data from UEs within a specific area (or outside a specific area).

As a result, UEs M1 and M2 are restricted to transfer their data to agent 1.

While transferring learning data to agent A2, UE M3 moves into a different area where 5G system provides prior notification on possible traffic congestion which might disturb UE M3 from keeping transferring learning data to agent A2.

UE M3 makes a selection of an action policy: (action 1) to defer the transfer or (action 2) to request to speed up the transfer.

* If action 1 is selected, UE M3 will resume transferring when it becomes available (not the main focus of this use case);
* If action 2 is selected, UE M3 will be able to (2a) more urgent/useful segment of data (send priority one over the other) and/or (2b) request more network resources to use.

*Modified connections (UEs M1 and M2):*

UE M1 attempts to get connected to agent A2 based on the information that agent A1 has provided when initially connected.

* UE M1 keeps transferring the data to agent A2
* Agent A1 can share the collected data or its processed form of data, vice versa

UE M2 is connected to agent A2

* UE M1 keeps transferring the data to agent A2
* Agent A1 can share the collected data or its processed form of data, vice versa

UE M3 is allowed to use more network resources to speed up the transfer to agent A2 (case 2b).



Figure 7.3.3-1: Modified connections b/w mobile devices and agents

### 7.3.4 Post-conditions

NOTE: Post-conditions are described in two different aspects (communication aspect and ML operation aspect).

Table 7.3.4-1: Post-conditions in communication aspect and ML operation aspect.

|  |  |  |
| --- | --- | --- |
| **Post-condition** | **Description in Communication aspects** | **Description in AI/ML operation aspects** |
| #1 | UE M1 and UE M2 can continue to transfer data. | Agent A1 and Agent A2 could continue to improve the outcome (e.g., learning model) even when disturbance happens. |
| #2 | UE M3 can (1) send priority portion of data over the other and/or (2) complete the transfer or maximize the transfer completion ratio *before congestion happens*. | Agent A2 could minimize the impact of temporal traffic congestion as UE M3 (1) transferred priority portion of data and (2) transferred more data to Agent A2 before traffic congestion happens and disturbs. |

### 7.3.5 Existing features partly or fully covering the use case functionality

Editor’s note: text to be provided.

### 7.3.6 Potential New Requirements needed to support the use case

In Table 7.3.6-1, the service-level implications of AI/ML operation are summarized in the third column of Table 7.3.6-1, which are used to derive high-level potential service requirements in communication layer in order to support the AI/ML operation.

Table 7.3.6-1: High level service requirements (in Communication and AI/ML operation aspects)

|  |  |  |
| --- | --- | --- |
| **PR** | **Potential Requirements in Communication aspects** | **Implications in AI/ML operation aspects (see note 1)** |
| [PR.7.3-001]  (see note 2) | 5G system shall provide a suitable means for a learning agent of the AI/ML operations service to use when providing a 5GS using AI/ML application with information (e.g., required increase in data rate due to changes in AI/ML operation, etc. requested by AI/ML application (e.g., of the learning agent or of UE)) necessary to minimize or avoid the impact of communication disruption on AI/ML service/performance.  Editor’s Note: Further performance requirements are FFS. | From this Use Case, the relevant AI/ML aspect is “A learning agent is able to provide a UE with information on candidate participating learning agents.” The above aspect can be supported by application layer above 3GPP transport layer. However, when transferring the learning data (processed, unprocessed or half-processed), if there are some changes in AI/ML layer to satisfy (e.g., triggered by some cause (e.g., due to technical or jurisdiction issue), it is intended that 3GPP help minimize or avoid the service disruption. |
| [PR.7.3-002] | 5G system shall provide a suitable means for a UE to provide the AI/ML application of a learning agent with information (e.g., experienced data rate, communication disturbance, geographical location to the extent of jurisdiction (e.g., State level, country level), etc.) necessary to minimize or avoid the impact of communication disruption on AI/ML service/performance.  Editor’s Note: Further performance requirements are FFS | From this Use Case, the relevant AI/ML aspect is “A UE is able to inform a learning agent of learning task splitting point.”  In the formulation of communication aspect, it is necessary that a collecting agent (UE in this case) provide some information for AI/ML application at the counterpart (which is the learning agent in the cloud) so that the learning agent can perform necessary actions in order to avoid or minimize the AI/ML service disruption. For this to happen, it is necessary that certain information in 5GS (UE) side be shared the learning agent in the cloud. |
| [PR.7.3-003]  (see note 3) | 5G system shall provide a means to supply prediction info (e.g., on traffic congestion, the related geographical area/spot) so that a UE or learning agent can minimize the impact of learning data transfer disturbance. |  |
| [PR.7.3-004]  (see note 4)  (see note 5)  [PR.7.3-004a] | 5G system shall be able to support adaptive scheduling (e.g., for network resource utilization for the communication between the 5G system and AI/ML application of a learning agent) according to increase or decrease of AI/ML traffic demand due to switching to a different AI/ML mode.  Editor’s Note: the above PR is FFS.  The 5G system shall be able to obtain from the AI/ML application of a learning agent the information about traffic demand increase or decrease (e.g. start/stop task splitting, change task splitting point) by that application (if available). | From this Use Case, the relevant AI/ML aspect is “The learning agent in the cloud needs to be aware of data rate change (e.g., traffic demand increase or decrease) that has been caused or will be caused by changes in workload sharing such as start using or stop using task splitting, or changing task splitting point that can increase or decrease the workload to UE as a collecting agent, in other words, decrease or increase the traffic demand)”. And this needs to be shared with 5GS in order for 5GS to be able to schedule efficiently. |
| [PR.7.3-xxy] |  | From this Use Case, what is required in AI/ML aspect is “The learning agent in the cloud or the learning agent in a UE is able to be aware of the disturbance that has happened in the counterpart (i.e., entity/UE that was transferring learning-related data to the learning agent).” |
| NOTE 1: From this Use Case, the AI/ML aspects/implications are summarized based on which the PR in the left-hand column is formulated  NOTE 2: It is one of possible scenarios that the learning agent can be located within MNO’s network (e.g., for learning-based signal processing optimization for RAN entity (e.g., gNB)), which operation is outside the scope of 3GPP but is used for optimization in RAN operations. It is intended that some information (e.g., QoS monitoring info) becomes available for 5G system so that 5G system can take necessary actions (e.g., “impact of communication disruption” for heavy data transfer can be minimized or avoided, which are service disruption in real-world applications, such as basic service robots). It is not intended to violate the QoS level (e.g., 5QI) when increasing or decreasing data rate.  NOTE 3: Disturbance by both jurisdiction (e.g., regional laws that prohibit personal data from being transferred) or by technical difficulty (e.g., traffic congestion for transferring heavy data for AI/ML) are considered in prediction.  NOTE 4: The intended scope of optimizing the network resources includes: to minimize the performance degradation, to minimize temporary imbalance among multiple AI/ML sessions that have changes in their data rate demand due to switching to a different mode (e.g., start using task splitting or stop using task splitting due to some communication disturbance or release from such disturbing factors, respectively) of AI/ML learning operation for a given model.  NOTE 5: the learning agent described in this requirement is in the cloud. | | |

Table 7.3.6-2 provides a summary of performance requirements for different usage scenarios. The required KPIs are dependent upon usage scenarios, especially on the task splitting points of given tasks even under the same usage scenarios. The calculation procedure can be referred to in [36-39].

For an example of image size 32 x 32 x 3 (32 wide, 32 high, 3 depth/colour channels), the weight is 3072; for images with more respectable size 200 x 200 x 3 = 120,000 weights; For a simple ConvNet for CIFAR-10 classification, the regular Neural Network architecture is INPUT🡪CONV🡪RELU🡪POOL🡪FC (Input layer, convolutional layer, pooling layer and fully-connected layer).

EXAMPLE 1: 32 x 32 x 3 image and six 5 x 5 filters produce a new image of size 28 x 28 x 6! = 564,480

EXAMPLE 2 (Language understanding): BERT\_{base} with L = 12 (layers), H = 768 (hidden size), A = 12 (heads). The number of parameters = 110M

EXAMPLE 3 (Language understanding): BERT\_{large} with L = 24 (layers), H = 1024 (hidden size), A = 16 (heads). The number of parameters = 340M

EXAMPLE 4: [39] for 8-bit VGG16 Pruned, it can reduce the original size (VGG-16 Ref) of 138MB by a factor of (1/13), which size will be approximately 10.3MB. Thus, 10.3MB / (GPU time / 2) ≒ 196MB/sec = 1.56Gb/sec.

NOTE 2: Compared to raw data, the latency requirement for trained data is considered more rigorous as it belongs to the category of data that is more readily usable by the related machine (e.g., by UE, or by agent).

Table 7.3.6-2: Performance requirements (KPI vectors)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description in Communication aspects** | | | **Description in AI/ML operation aspects (all inclusive)** | | | **Range (see note 2)** |
| **End-to-end latency (see note 1)** | **User experienced data rate (see note 1)** | **Service interruption time** | **End-to-end latency** | **Data rate** | **Service interruption time** |
| Learning data (raw data) at Service robot at initial launch  (see note 4) | < 100 ms | UL: < [1.5] Gbit/s  DL: < [500] Mbit/s |  | < 10 s |  | < [100] ms | < [500] km |
| Learning data (trained data) at Service robot at initial launch  (see note 4) | < 50 ms | UL: < [50] Mbit/s  DL: < [500] Mbit/s |  | < 10 s |  | < [100] ms | < [500] km |
| Learning data (raw data) at Service robot in routine operation  (see note 4) | < 100 ms | UL: 700 Mbit/s  DL: < [500] Mbit/s |  | < 1 s |  | < [100] ms | < [200] km |
| Learning data (trained data) at Service robot in routine operation  (see note 4) | < 50 ms | UL: 50 Mbit/s  DL: < [500] Mbit/s |  | < 1 s |  | < [100] ms | < [200] km |
| Remote control of robots (type 1) (see note 3) | < [5] ms | UL: < [700] Mbit/s  DL: [1] Mbit/s |  | < [10] ms |  | < [10] ms | < [3] km  (aerial, outdoor);  < [500] m (indoor) |
| Remote control of robots (type 2) (see note 3) | < [20] ms | UL: < [700] Mbit/s  DL: [1] Mbit/s |  | < [25] ms |  | < [100] ms | < [3] km |
| NOTE 1: The end-to-end latency and user experienced data rate are dependent on which learning model the UE and agent have selected to use. The expected user experienced data rate varies depending on the task splitting point between UE and the learning agent. The suggested number in this table is based on the maximum possible value (e.g., non-splitting cases or on splitting stage at pooling).  NOTE 2: The range is a rough measure, which may include radio and core network segments, between a Service Robot (as a UE) and the learning agent (including candidate learning agent). For routing operation, the range is typically smaller than that for initial launch of a task/job site.  NOTE 3: For robot control, only two categories are suggested. A refined use of more categories is FFS. Type 1 requires more rigorous KPIs (e.g., helicopter, humanoid robots). This case is not intended to cover more rigorous and/or complicated operational environments such as a use case with a very long communication range and tactile/haptic feedback related use cases demanding ultra-fast delivery of skillsets, e.g., in (or less than) the order of milliseconds.  NOTE 4: In these scenarios, service robots are assumed to have basic robotics mobility. DL (downlink) data rate requirement is characterised by the required upper bound that can often happen sporadically, e.g., when downloading AI/ML models. | | | | | | | |

## 7.4 Group Performance “Flocking” Use Case

### 7.4.1 Description

A new ‘service enabler’ is introduced that allows a service provider to achieve effective performance for the entire group of devices. The term ‘Flock’ stands for a group that has performance requirements that consider the performance ‘as a team’ as opposed to the ‘total’ or results of the ‘best performers.’

The example of a Flock provided in this use case (an application of this service enabler) is for Synchronous Federated Learning. Please note that “Flocking” is a **general service enabler.** Synchronous Federated Learning is discussed here because (a) it is a concrete application of the “Flocking” service enabler, (b) the topic of this study is model and AI data set communication. To explain and motivate “Flocking” we need an example, and fortunately this application is directly within the scope of the FS\_AMMT study.

Synchronous federated learning involves a set of contributing terminals, as described in clause 7 of this TR. In a federation, a hierarchy exists that provides an effective delegation of work and information. This federation functions as if it were a single (non-federated) system to the extent that the distributed components can operate within the same expectations. For synchronous federated learning some number of the federation’s components lag, these become stragglers. Information and function availability of the whole federation suffers when the performance of individual components fall significantly behind the others as the entire group should complete the iteration.

Synchronous federated learning works best by eliminating bias – allowing diverse users and devices to participate and bring to the learning task diversity of input data, as the users will have different attributes. It is important not to merely focus on the ‘best performing devices’ in the federation and drop the rest. It may increase the performance in terms of time to iterate the synchronous federated learning task to drop stragglers, but this will reduce the diversity of the data set and introduce bias.

Where group performance is defined by the weakest member (as in the slowest flying bird), we term this a “flock.” The 5GS normally considers performance objectives and QoS for individual communicating terminals. Here, the 5GS QoS objective relates to the entire set of terminals making up the federation, the “flock” of UEs.

### 7.4.2 Pre-conditions

A set of UEs that participated in federated learning exists. These UEs have registered with a PLMN and operate in a federation to perform federated learning tasks.

The federated learning service provider “Avian” organizes the work of these UEs so that repeated iterations of training will occur over time.

It is assumed that the UEs provide federated learning input using the same network resources (e.g. network slice) and that the policy for this network communication is distinct from the policy for other activities that the UE performs. In this way, the network can adjust the QoS policy for federated learning communication for individual UEs without any service impact except to the federated learning service.

### 7.4.3 Service Flows

As the performance and quality of the output of the entire set of UEs is bounded by the performance of the weakest members of the group, Avian provides the 5GS with a policy identifying the reporting interval for which different iterations should conclude. Avian also provides reports on the progress of different UEs as they proceed. The 5GS is then in a position to adjust the QoS policies of some UEs *to allocate more resources* for those UEs that lag, and *less resources* for those that are ahead of the flock. Therefore, the slowest UEs (e.g. at producing a report after an iteration of a federated learning task) achieve an improved performance. The fastest UEs (e.g. also at producing a report after a federated learning task) do not need as much network resources (higher QoS), so the 5GS can reduce the QoS guarantees for these, and thereby saves these resources. The overall result is more efficient for the Synchronous Federated Learning service and for the network operator. The resource allocated to UE is maintained for at least one iteration.

The 5GS can inform Avian of any additional UEs with good communication performance (e.g. due to radio resources) and/or existing UEs whose connection has degraded to a level which is no longer sufficient for FL tasks. This enables Avian to determine when to add new UEs into the flock or remove existing UEs from the flock.

NOTE: While it is clear that the speed with which training occurs and reports are generated by UEs is only partially bounded by communication, it is assumed that the communication resources available to the UE is a significant contributor to the time it requires to complete a training iteration.

When a new UE joins the federation, it will register with Avian. Avian can then notify the 5GS (by means of a standard interface) of this addition. This interface is depicted logically in Figure 7.4.3-1 below.

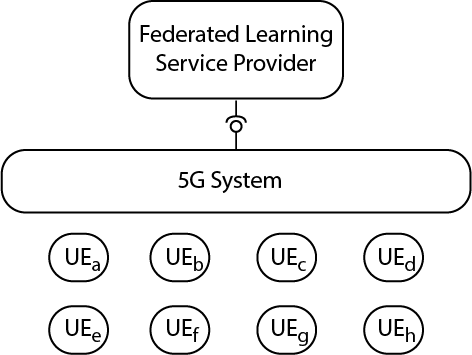


Figure 7.4.3-1: 5G Service Enabler interface, Example for Synchronous Federated Learning

Similarly, when a UE leaves the federation, the 5GS is notified. This allows the 5GS to modify the policy to balance the QoS policy to achieve the most consistent performance across the involved UEs. During the adjustment of QoS policy, the total communication resource (e.g. total GBR of all members in the flock) can be given a maximum set of resources, (e.g. a GBR aggregate that should not exceed a maximum value).

As the data transmission for Federated Learning is not for regular data services such as video, voice call, webpage browsing, etc., the 5GS needs to have a charging exemption or rewarding to this kind of traffic.

### 7.4.4 Post-conditions

The ‘flock’ of UEs performs consistently.

### 7.4.5 Existing features partly or fully covering the use case functionality

The existing QoS features controlled by the network with reconfigurable policy provide necessary but not sufficient functionality to support the use case.

### 7.4.6 Potential New Requirements needed to support the use case

[P.R.7.4-001] The 5G system shall be able to support ‘aggregated performance’ for a group of UEs where the worst performing member defines the performance of the entire group. E.g. the 5G system could achieve performance for the entire group so as to avoid members achieving either significantly less or more performance than others in the group.

[P.R.7.4-002] The 5G system shall be able determine whether a required QoS for each member in a group can be maintained.

[P.R.7.4-003] The 5G system shall be able to expose QoS information for a group of UEs to an authorized service provider.

Editor’s Note: the requirement above is FFS.

[P.R.7.4-004] The 5G core network shall support collection of charging information based on whether the traffic is for AI/ML services.

Editor’s Note: it is to check whether it is already covered by existing requirement

# 8 Consolidated potential requirements

### 8.1 KPI for AMMT services

Editor’s Note: This section summarizes the available KPI values from the various approved use cases based on v0.3.0. The KPI values in the tables may be corrected/modified aligned with the newly-added use-cases and the updates to the approved use-cases.

In table 8.1-1, 8.1-2 and 8.1-3, only the subset of experienced data rate values which are agreeable for the expected 5G enhancements are captured from the use cases.

* The agreeable upper bound for split AI/ML inference is [1.1Gbit/s] in DL and UL respectively.
* The agreeable upper bound for AI/ML model downloading is [1.1Gbit/s] in DL. The upper bound is [4Gbit/s] in DL if the downloading is only supported in hotspot coverage.
* The agreeable upper bound for Federated Learning is [1.1Gbit/s] in DL and UL respectively.

NOTE: It might be possible in the selection of members for the federated/distributed learning to prioritize users in good coverage areas. In that case, higher data rates can be possible.

Table 8.1-1 KPI Table of split AI/ML inference between UE and Network Server/Application function

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Uplink KPI** | | | | | **Downlink KPI** | | | | **Remarks** |
| **Max allowed UL end-to-end latency** | **Experienced data rate** | **Payload size** | **Communication service availability** | **Reliability** | **Max allowed DL end-to-end latency** | **Experienced data rate** | **Payload size** | **Reliability** |
| [CPR-001] 2ms | [CPR-002] 1.08Gbit/s (see note 1) | 0.27 MByte | [CPR-003] 99.999 % | [CPR-037] 99.9% |  |  |  | [CPR-038] 99.999% | Split AI/ML image recognition |
| [CPR-004] [100ms] | [CPR-005] [1.5Mbit/s] |  |  |  | [CPR-006] [100ms] | [CPR-007] [150] Mbit/s | 1.5 MByte /frame |  | Enhanced media recognition |
|  |  | 4.7Mbit/s |  |  | [CPR-008]  12ms | [CPR-009]  320Mbit/s | 40kByte |  | Split control for robotics |
| NOTE 1: Only the values corresponding to AlexNet model is captured.  NOTE 2: Communication service availability relates to the service interfaces, and reliability relates to a given system entity. One or more retransmissions of network layer packets may take place in order to satisfy the reliability requirement. | | | | | | | | | |

Table 8.1-2 KPI Table of AI/ML model downloading

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Max allowed DL end-to-end latency** | **Experienced data rate**  **(DL)** | **Model size** | **Communication service availability** | **Reliability** | **User density** | **# of downloaded AI/ML models** | **Remarks** |
| [CPR-010]  1s | [CPR-011]  1.1Gbit/s | 138MByte | [CPR-012] 99.999 % | [CPR-039] 99.9% for data transmission of model weight factors; 99.999% for data transmission of model topology |  |  | AI/ML model distribution for image recognition |
| [CPR-013]  1s | [CPR-014]  640Mbit/s | 80MByte | [CPR-015] 99.999 % |  |  |  | AI/ML model distribution for speech recognition |
| [CPR-016]  1s | [CPR-017]  512Mbit/s / [4Gbit/s]  (see note 1) | < 64MByte / 500MByte |  |  |  | Parallel download of up to 50 AI/ML models | Real time media editing with on-board AI inference |
| [CPR-018]  1s |  | 536MByte |  |  | [CPR-019]  up to 5000~ 10000/km2 in an urban area |  | AI model management as a Service |
| [CPR-020]  [500ms] | [CPR-021]  [100 Mbit/s] | [40MByte] | [CPR-022] 99.999 % |  |  |  | AI/ML based Automotive Networked Systems |
| [CPR-023]  [1s] |  | [500]MByte |  |  |  |  | Shared AI/ML model monitoring |
| [CPR-024]  3s | [CPR-025]  450Mbit/s | [CPR-026]  170MByte |  |  |  |  | Media quality enhancement |
| NOTE 1: 512Mbit/s concerns AI/ML models having a size below 64 MB. 4Gbit/s concerns AI/ML models having a size below 500 MB where the model downloading is only supported in hotspot coverage.  NOTE 2: Communication service availability relates to the service interfaces, and reliability relates to a given system entity. One or more retransmissions of network layer packets may take place in order to satisfy the reliability requirement. | | | | | | | |

Table 8.1-3 KPI Table of Federated Learning between UE and Network Server/Application function

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Max allowed DL or UL end-to-end latency** | **DL experienced data rate** | **UL experienced data rate** | **DL packet size** | **UL packet size** | **Communication service availability** | **Remarks** |
| [CPR-027]  [1]s | [CPR-028]  1.0Gbit/s | [CPR-029]  1.0Gbit/s | 132MByte | 132MByte |  | Uncompressed Federated Learning for image recognition |
| [CPR-030]  [1s] | [CPR-031]  80.88Mbit/s | [CPR-032]  80.88Mbit/s | 10Mbyte | 10Mbyte | [CPR-033]  [99.9%] | Compressed Federated Learning for image/video processing |
| [CPR-034]  [1s] | [CPR-035]  [1.1Gbit/s] | [CPR-036]  [500Mbit/s] | 10MByte | 10MByte |  | Data Transfer Disturbance in Multi-agent multi-device ML Operations |

### 8.2 Functional requirements for AMMT services

Editor’s Note: This section summarizes the available requirements from the various approved use cases based on v18.1.0. The requirements in the tables may be corrected/modified aligned with the newly-added use-cases and the updates to the approved use-cases.

Table 8.2-1 Functional requirements for AMMT services

| CPR # | Consolidated Potential Requirement | Original PR # | Comment |
| --- | --- | --- | --- |
| CPR 8.2-1 | Based on operator policy, the 5G network shall provide the means to allow an authorized third-party to monitor the resource utilisation of the network service that is associated with the third-party.  NOTE: Resource utilization in the preceding requirement refers to measurements relevant to the UE’s performance such as the data throughput provided to the UE. | PR.5.5-001 |  |
| CPR 8.2-2 | Based on operator policy, the 5G system shall be able to provide an indication about a planned change of bitrate, latency, or reliability for a QoS flow to an authorized 3rd party so that the 3rd party AI/ML application is able to adjust the application layer behaviour if time allows. The indication shall provide the anticipated time and location of the change, as well as the target QoS parameters. | PR.5.5-002 |  |
| CPR 8.2-3 | Based on operator policy, 5G system shall be able to provide the means to predict (to the extent possible) and expose predicted network condition changes (i.e. bitrate, latency, reliability) per UE to the authorized third party. | PR.5.5-003 |  |
| CPR 8.2-4 | Subject to user consent, operator policy and regulatory constraints, the 5G system shall support a mechanism to expose monitoring and status information of an AI-ML session, (e.g. measured data rate/delay and other traffic analytics information), to a 3rd party AI/ML application.  NOTE: such mechanism is needed for AI/ML application to determine an in-time transfer of AI/ML model | PR.6.4-001  PR.6.7-001 | The two PRs are proposing to make 5GS provide information including measured data rate, delay, analytics result, and prediction for communication to 3rd party, for AI model downloading and training.  It is proposed to merge the two PRs |
| CPR 8.2-5 | 5G system shall provide a means to supply event alerting to an authorized 3rd party, together with a predicted time of the event. (e.g., alerting about traffic congestion or UE moving into/out of a different geographical area).  NOTE: A 3rd party AI/ML application may use the prediction information to minimize disturbance in the transfer of learning data and AI/ML model data. | PR.7.3-003 | It is proposed to adopt it with the modification “learning agent –> 3rd party” and rewording |
| CPR 8.2-6 | The 5G system shall be able to support an authorised 3rd party to change aggregated QoS parameter values associated with a group of UEs, e.g. UEs of a FL group. | PR 7.4-001 |  |
| CPR 8.2-7 | Subject to user consent, operator policy and regulatory requirements, the 5G system shall be able to expose information to an authorized 3rd party to support the 3rd party to determine members of a group of UEs, e.g. UEs of a FL group, based upon criteria provided in the request from the 3rd party. | PR 7.4-001 |  |
| CPR 8.2-8 | The 5G system shall be able to expose aggregated QoS parameter values for a group of UEs to an authorized service provider. | PR.7.4-002  PR.7.4-003  PR.7.4-004 | It is proposed to adopt the three potential requirements into CPRs |
| CPR 8.2-9 | The 5G system shall be able to support collection of charging information for a group of UEs, e.g. UEs of a AI/ML FL group. | PR 7.4-004 | The CPR is changed from the original PR to support the charging for FL group where the member may be dynamically changed. |

# 9 Conclusion and recommendations

Regarding the Feasibility Study on traffic characteristics and performance requirements for AI/ML model transfer in 5GS, the TR analyses use cases of AMMT as follows:

* Use cases on AI/ML operation splitting between AI/ML endpoints:
  + Split AI/ML image recognition;
  + Enhanced media recognition: Deep Learning Based Vision Applications;
  + Media quality enhancement: Video streaming upgrade;
  + Split control for robotics;
  + Session-specific model transfer split computation operations.
* Use cases on AI/ML model/data distribution and sharing over 5G system:
  + AI/ML model distribution for image recognition;
  + Real time media editing with on-board AI inference;
  + AI/ML model distribution for speech recognition;
  + AI model management as a Service;
  + AI/ML based Automotive Networked Systems;
  + Shared AI/ML model monitoring;
  + Prediction of AI/ML model distribution.
* Use cases on Distributed/Federated Learning over 5G system:
  + Uncompressed Federated Learning for image recognition;
  + Compressed Federated Learning for image/video processing;
  + Data Transfer Disturbance in Multi-agent multi-device ML Operations;
  + Group Performance “Flocking” Use Case.

It is recommended to proceed with normative work, and include the potential new requirements identified by this TR into a new TS. The consolidated potential requirements in Clause 8 are candidates for the normative requirements.

Annex A:  
Introduction to AI/ML models

## A.1 AI and ML

Artificial Intelligence (AI)/Machine Learning (ML) is being used in a range of application domains across industry sectors, realizing significant productivity gains. In particular, in mobile communications systems, mobile devices (e.g. smartphones, smart vehicles, UAVs, mobile robots) are increasingly replacing conventional algorithms (e.g. speech recognition, machine translation, image recognition, video processing, user behaviour prediction) with AI/ML models to enable applications like enhanced photography, intelligent personal assistants, VR/AR, video gaming, video analytics, personalized shopping recommendation, autonomous driving/navigation, smart home appliances, mobile robotics, mobile medicals, as well as mobile finance. As forecast by Gartner [44], more than 80% of enterprise IoT projects will include an AI component by 2022, up from only 10% today.

***Artificial Intelligence (AI)*** is the science and engineering to build intelligent machines capable of carrying out tasks as humans do, defined by John McCarthy in 1956. The categorization of AI approaches can be illustrated in figure A.1-1 [25].

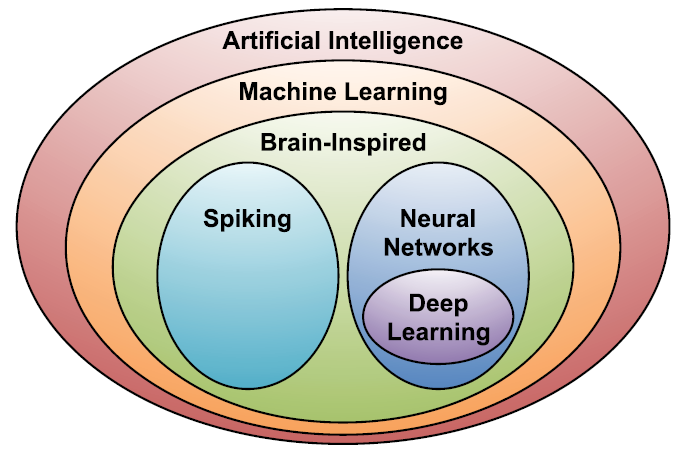


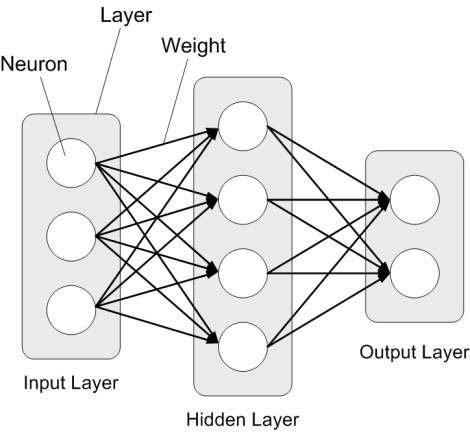
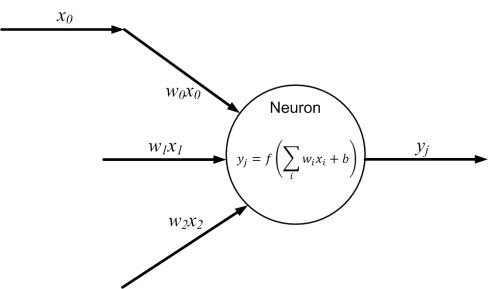
Figure A.1-1. Categorization of AI/ML approaches (figure adopted from [25])

Within AI is a large subfield called ***machine learning (ML)***, which was defined in 1959 by Arthur Samuel as the field of study that gives computers the ability to learn without being explicitly programmed. Instead of the laborious and hit-or-miss approach of creating a distinct, custom program to solve each individual problem in a domain, a single ML algorithm simply needs to learn, via a process called training, to handle each new problem [25]. Many ML methodologies as exemplified by decision tree, K-means clustering, and Bayesian network have been developed to train the model to make classifications and predictions, based on the data obtained from the real world [19].

## A.2 Deep neural network

Within the ML field, there is an area that is often referred to as brain-inspired computation, which is a program aiming to emulate some aspects of how we understand the brain to operate. Since it is believed that the main computational elements a human brain are 86 billion neurons, the two subareas of brain-inspired computation are both inspired by the architecture of a neuron [25], as shown in figure A.2-1 (a).

Compared to spiking computing approaches, e.g. [3], the more popular ML approaches are using ***“neural network”*** as the model. Neural networks (NN) take their inspiration from the notion that a neuron’s computation involves a weighted sum of the input values. But instead of simply outputting the weighted sum, a NN applies a nonlinear function to generate an output only if the inputs cross some threshold, as shown in figure A.2-1(a). Figure A.2-1(b) shows a diagrammatic picture of a computational neural network. The neurons in the input layer receive some values and propagate them to the neurons in the middle layer of the network, which is also called a “hidden layer”. The weighted sums from one or more hidden layers are ultimately propagated to the output layer, which presents the final outputs of the network [25].



1. (b)

Figure A.2-1. Architecture of neuron and neural network

Neural networks having more than three layers, i.e., more than one hidden layer are called ***deep neural networks (DNN)***. In contrast to the conventional shallow-structured NN architectures, DNNs, also referred to as deep learning, made amazing breakthroughs since 2010s in many essential application areas because they can achieve human-level accuracy or even exceed human accuracy. Deep learning techniques use supervised and/or unsupervised strategies to automatically learn hierarchical representations in deep architectures for classification [26]. With a large number of hidden layers, the superior performance of DNNs comes from its ability to extract high-level features from raw sensory data after using statistical learning over a large amount of data to obtain an effective representation of an input space [25]. In recent years, thanks to the big data obtained from the real world, the rapidly increased computation capacity and continuously-evolved algorithms, DNNs have become the most popular ML models for many AI applications.

## A.3 Training and inference

***Training*** is a process in which a AI/ML model learns to perform its given tasks, more specifically, by optimizing the value of the weights in the DNN. A DNN is trained by inputting a training set, which are often correctly-labelled training samples. Taking image classification for instance, the training set includes correctly-classified images. When training a network, the weights are usually updated using a hill-climbing optimization process called gradient descent. The gradient indicates how the weights should change in order to reduce the loss (the gap between the correct outputs and the outputs computed by the DNN based on its current weights). The training process is repeated iteratively to continuously reduce the overall loss [25]. Until the loss is below a predefined threshold, the DNN with high precision is obtained.

There are multiple ways to train the network for different targets. The introduced above is supervised learning which uses the labelled training samples to find the correct outputs for a task. Unsupervised learning uses the unlabelled training samples to find the structure or clusters in the data. Reinforcement learning can be used to output what action the agent should take next to maximize expected rewards. Transfer learning is to adjust the previously-trained weights (e.g. weights in a global model) using a new training set, which is used for a faster or more accurate training for a personalized model [25].

After a DNN is trained, it can perform its task by computing the output of the network using the weights determined during the training process, which is referred to as ***inference***. In the model inference process, the inputs from the real world are passed through the DNN. Then the prediction for the task is output, as shown in Figure A.3-1. For instance, the inputs can be pixels of an image, sampled amplitudes of an audio wave or the numerical representation of the state of some system or game. Correspondingly, the outputs of the network can be a probability that an image contains a particular object, the probability that an audio sequence contains a particular word or a bounding box in an image around an object or the proposed action that should be taken [25].

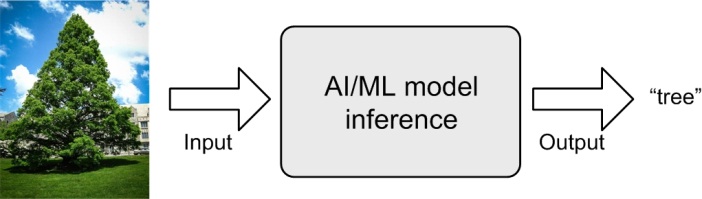


Figure A.3-1. Example of AI/ML inference

The performance of DNNs is gained at the cost of high computational complexity. Hence more efficient compute engines are often used, e.g. graphics processing units (GPU) and network processing units (NPU). Compared to the inference which only involves the feedforward process, the training often requires more computation and storage resources because it involves also the backpropagation process [10].

## A.4 Widely-used DNN models and algorithms

Many DNN models have been developed over the past two decades. Each of these models has a different “network architecture” in terms of number of layers, layer types, layer shapes (i.e., filter size, number of channels and filters), and connections between layers [25]. Figure A.4-1 presents three popular structures of DNNs: multilayer perceptrons (MLPs), convolution neural networks (CNNs), and recurrent neural networks (RNNs). ***Multilayer perceptrons (MLP)*** model is the most basic DNN, which is composed of a series of fully connected layers [41]. In a fully connected layer, all outputs are connected to all inputs, as shown in Figure A.4-1. Hence MLP requires a significant amount of storage and computation.

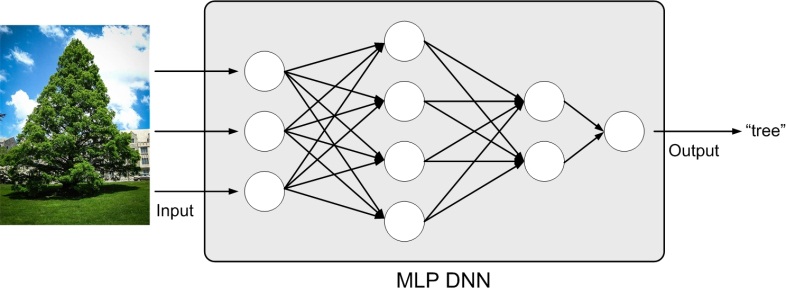


Figure A.4-1. MLP DNN model

An approach to limiting the number of weights that contribute to an output is to calculate the output only using a function of a fixed-size window of inputs. An extremely popular window-based DNN model uses a convolution operation to structure the computation, hence is named as ***convolution neural network (CNN)*** [25]. A CNN is composed of multiple convolutional layers, as shown in figure A.4-2. Applying various convolutional filters, CNN models can capture the high-level representation of the input data, making it popular for image classification [7] and speech recognition [42] tasks. In recent years, the modern CNN models have dramatically improved the performance of image classification tasks (e.g., AlexNet [7], VGG network [8], GoogleNet [9], ResNet [18], MobileNet [19]), as shown in figure A.4-3 [25].

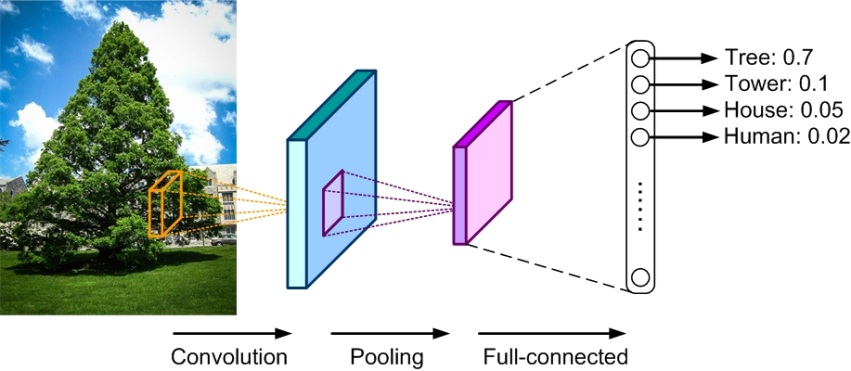


Figure A.4-2. CNN model

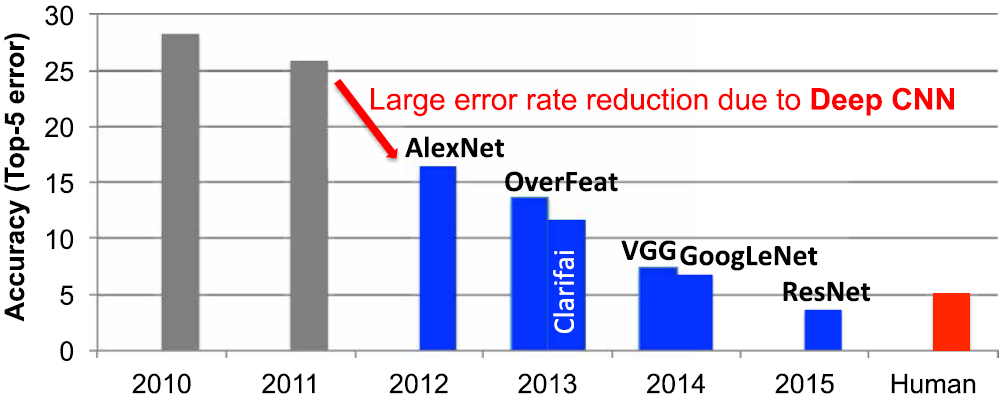


Figure A.4-3. Image classification improvements made by CNN models (Figure adopted from [25])

***Recurrent neural network (RNN)*** models are another type of DNNs, which use sequential data feeding. The input of RNN consists of the current input and the previous samples. Each neuron in an RNN owns an internal memory that keeps the information of the computation from the previous samples. As shown in figure A.4-4, the basic unit of RNN is called cell, and further, each cell consists of layers and a series of cells enables the sequential processing of RNN models. RNN models have been widely used in the natural language processing task on mobile devices, e.g., language modelling, machine translation, question answering, word embedding, and document classification.



Figure A.4-4. RNN model

***Deep reinforcement learning (DRL)*** is not another DNN model. It is composed of DNNs and reinforcement learning [43]. As illustrated in figure A.4-5, the goal of DRL is to create an intelligent agent that can perform efficient policies to maximize the rewards of long-term tasks with controllable actions. The typical application of DRL is to solve various scheduling problems, such as decision problems in games, rate selection of video transmission, and so on.

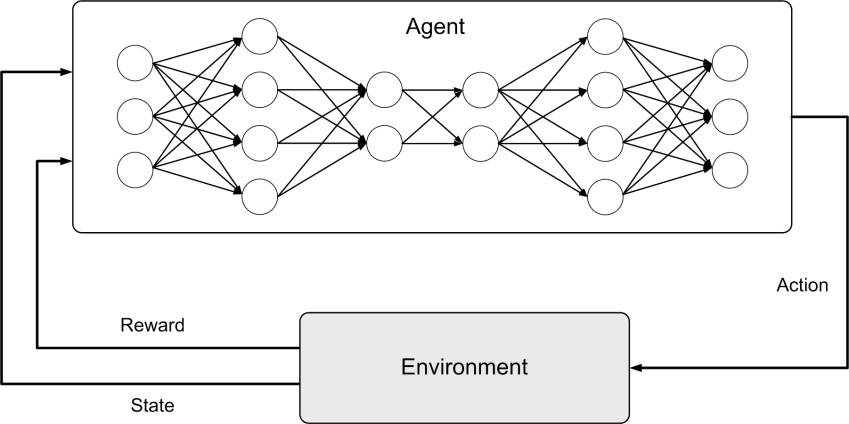


Figure A.4-5. Reinforcement learning

Annex B:  
General principle of split AI/ML operation between AI/ML endpoints

In recent years, the AI/ML-based mobile applications are increasingly computation-intensive, memory-consuming and power-consuming. Meanwhile end devices usually have stringent energy consumption, compute and memory cost limitations for running a complete offline AI/ML inference onboard. Hence many AI/ML applications currently intent to offload the inference processing from mobile devices to internet datacenters (IDC). Nowadays, even photos shot by a smartphone are often processed in a cloud AI/ML server before shown to the user who shot them. However, the cloud-based AI/ML inference tasks need to take the following factors into account:

1. Computation pressure at IDCs

As the estimates in [44], by 2021, nearly 850ZB data will be generated by end devices per year, whereas the global IDC traffic can only reach 20.6ZB. That means most of the data can only be left at network edge (i.e. devices and MEC) for AI/ML processing.

1. Required data rate and latency

Increasing number of AI/ML applications are requiring a high data rate meanwhile a low latency for communications between devices and the network, e.g. VR/AR, automatic driving, remote-controlled robotics. According to the estimates in [44] on device-initiated traffic, offloading all the data to cloud servers for AI/ML inference would consume excessive uplink bandwidth. This introduces challenging requirements on mobile communications system capacity, including for the 5G system.

1. Privacy protection requirement

The sensing/perception data supporting the inference in the cloud server often carry privacy of the end users. Different types of privacy protection problems need to be considered in case of either processing the data at the device or reporting it to the cloud/edge server. Compared to reporting it to the server, keeping the raw data at the device can reduce the pressure of privacy protection at the network side.

Hence in many cases, the split AI/ML inference over device and network are required, to enable the AI/ML applications with conflicting requirements which are computation-intensive, energy-intensive as well as privacy-sensitive and delay- sensitive. Many references [10-14] have shown that processing AI/ML inference with device-network synergy can alleviate the pressure of computation, memory footprint, storage, power and required data rate on devices, reduce end-to-end latency and energy consumption, and improve the end-to-end accuracy and efficiency when compared to the local execution approach on either side.

The scheme of split AI/ML inference can be depicted in Figure B.1-1. The AI/ML operation/model is split into multiple parts according to the current task and environment. The intention is to offload the computation-intensive, energy-intensive parts to network endpoints, whereas leave the privacy-sensitive and delay- sensitive parts at the end device. The device executes the operation/model up to a specific part/layer and sends the intermediate data to the network endpoint. The network endpoint executes the remaining parts/layers and feeds the inference results back to the device. It should be noted that, in the example in Figure B.1-1, the final inference result is output by network AI/ML endpoint 2. According to actual use case, the inference result can also be output by other endpoints, e.g. network AI/ML endpoint 1.

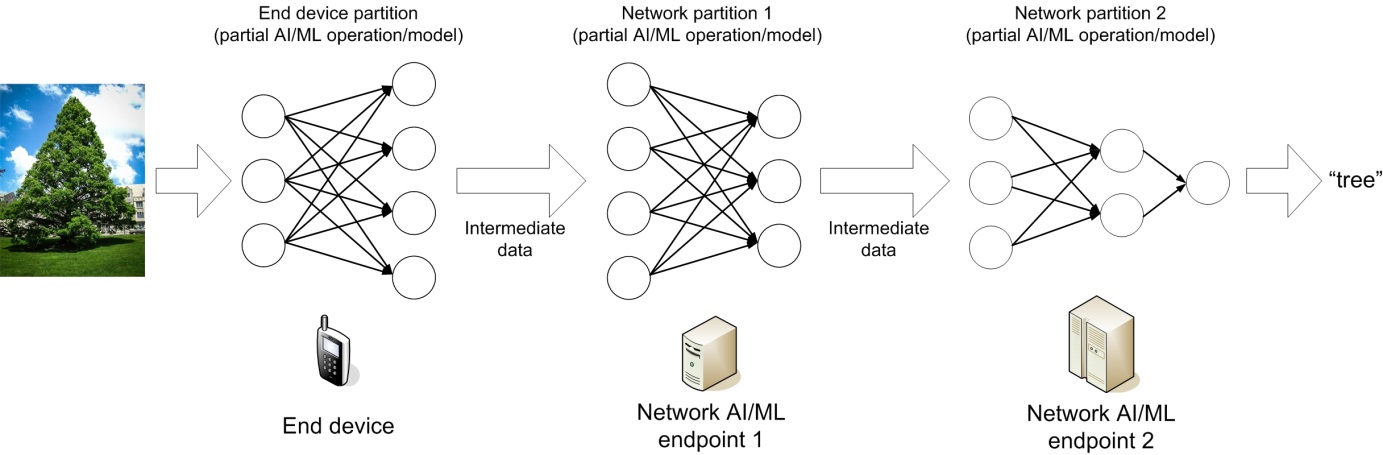


Figure B.1-1. Example of split AI/ML inference

The modes for split AI/ML operations between device and network are illustrated in Figure B.1-2. The modes are in general applicable for AI/ML training as well as inference. In this section, we focus on the inference processing. Mode a) and b) are traditional schemes operating the AI/ML inference wholly on one endpoint. Mode c) - g) attempt to split the AI/ML inference or even the model into multiple parts according to the current task and environment, to alleviate the pressure of computation, memory/storage, power and required data rate on both device and NW endpoints, as well as to obtain a better model inference performance on latency, accuracy and privacy protection.

* Mode a): Cloud/edge-based inference

In this mode (as shown in Figure B.1-2 (a)), the AI/ML model inference is only carried out in a cloud or edge server. The device only reports the sensing/perception data to the server, and does not need to support AI/ML inference operations. The server returns the inference results to the device. The advantage of this mode is limiting the device complexity. One disadvantage is that the inference performance depends on communications data rate and latency between the device and the server. Real-time uploading some perception data (e.g. high-resolution video streaming) requires a stably-high data rate and some AI/ML services (e.g. remote-controlled robotics) requires a stably-low latency, which are challenging to be guaranteed in 5G system due to different network coverages. And due to the disclosure of the privacy-sensitive data to the network, corresponding privacy protection measurements are required.

* Mode b): Device-based inference

In this mode (as shown in Figure B.1-2 (b)), the AI/ML model inference is performed locally at the mobile device. The advantage is that, during the inference process, the device does not need to communicate with the cloud/edge server. Another motivation of this mode is preserving the privacy at the data source, i.e. the device, although the privacy protection problem needs also be considered at the device side. The disadvantage is potentially imposing an excessive computation/memory/storage resource to the device. And also pointed out by [10], we cannot assume the device always keep all the potentially-needed AI/ML models onboard. In some cases, the mobile device may need to obtain the AI/ML model from the edge cloud/server, which requires a corresponding downloading data rate from the 5G system, as introduced in Section 7.

* Mode c): Device-cloud/edge split inference

In this mode (as shown in Figure B.1-2 (c)), an AI/ML inference operation or model is firstly split into two parts between the device and the cloud/edge server according to the current system environmental factors such as communications data rate, device resource, and server workload. Then, the device will execute the AI/ML inference up to a specific part or the DNN model up to a specific layer, and send the intermediate data to the cloud/edge server. The server will execute the remaining part/layers and sends the inference results to the device. Compared to Mode a) and b), this mode is more flexible and more robust to the varying computation resource and communications condition. A key link for this mode is to properly select the optimum split point between device side and network side based on the conditions.

* Mode d): Edge-cloud split inference

This mode (as shown in Figure B.1-2 (d)) can be regarded as an extension of Mode a). The difference is that the DNN model is executed through edge-cloud synergy, rather than executed only on either cloud or edge server. The latency-sensitive part of an AI/ML inference operation or layers of an AI/ML model can be performed at the edge server. The computation-intensive parts/layers that the edge server cannot perform can be offloaded to cloud server. The device only reports the sensing/perception data to the server, and does not need to support AI/ML inference operations. The intermediate data are sent from the edge server to the cloud server. A proper split point needs to be selected for an efficient cooperation between edge server and cloud server.

* Mode e): Device-edge-cloud split inference

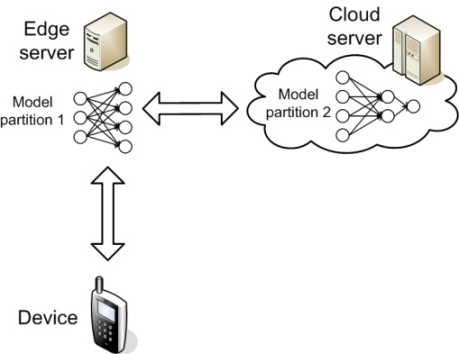
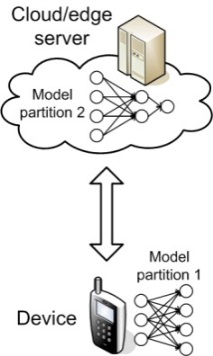
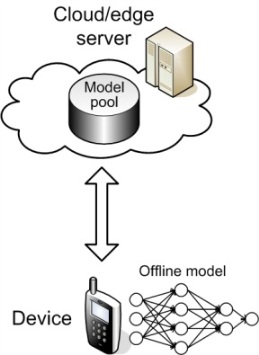
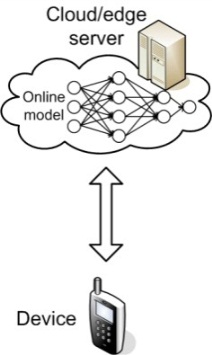
This mode (as shown in Figure B.1-2 (e)) is the combination of Mode c) and d). An AI/ML inference operation or an AI/ML model is split over the mobile device, the edge server and the cloud server. The computation-intensive parts/layers of an AI/ML operation/model can be distributed among the cloud and/or edge server. The latency-sensitive parts/layers can be performed on the device or the edge server. The privacy-sensitive data can be left at the device. The device sends the intermediate data outcome from its computation to the edge server. And the edge server sends the intermediate data outcome from its computation to the cloud server. Two split points need to be selected for an efficient cooperation between the device, the edge server and the cloud server.

* Mode f): Device-device split inference

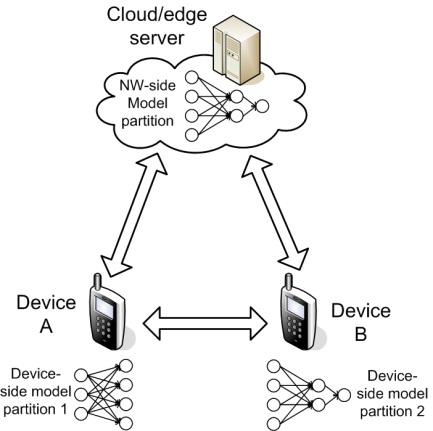
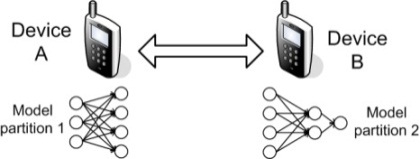
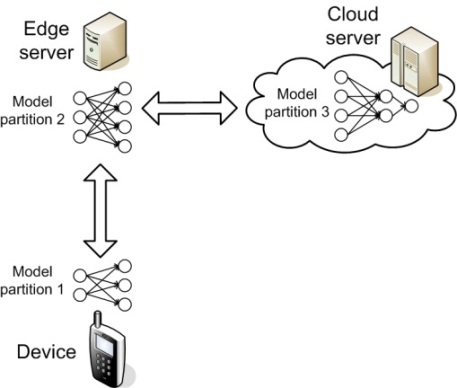
This mode (as shown in Figure B.1-2 (f)) provides a de-centralized split inference. An AI/ML inference operation or model can be split over different mobile devices. A group of mobile devices can perform different parts of an AI/ML operation or different DNN layers for an inference task, and exchange intermediate data between each other. The computation load can be distributed over devices meanwhile each device preserves it private information locally.

* Mode g): Device-device-cloud/edge split inference

Mode g) can be further combined with Mode c) or e). As shown in Figure B.1-2 (g), an AI/ML inference operation or model is firstly split into the device part and network part. Then the device part can be executed in a de-centralized manner, i.e. further split over different mobile devices. The intermediate data can be sent from one device to the cloud/edge server. Or multiple devices can send intermediate data to the cloud/edge server.



(a) (b) (c) (d)



(e) (f) (g)

Figure B.1-2. Split AI/ML inference modes over endpoints

Annex C:  
General principle of AI/ML model/data distribution and sharing over 5G system

For the inference tasks which requires low latency and desires the privacy-sensitive data to be preserved at the UE side, offline AI/ML inference is desired, rather than the cloud-based inference. However, an offline AI/ML model running on mobile devices must have a relatively low computation complexity and a small storage size. An approach to enabling offline DNN models on mobile devices is to compress the model to reduce its resource and computational requirements [27-28, 35, 45]. However, DNN compression will lead to loss of inference accuracy and adaptivity to various tasks and environments. A solution to this challenge is to adaptively select the model for inference from a set of trained models [10]. The model selection is motivated by the observation that the optimum model for inference depends on the input data and the precision requirement [20][45]. Multi-functional mobile terminals usually need to switch the AI/ML model in response to task and environment variations.

The condition of adaptive model selection is that the models to be selected have been available for the mobile device. However, given the fact that the DNN models are becoming increasingly diverse, and with the limited storage resource in a UE, it is unfeasible to pre-load all candidate AI/ML models on-board. Online model distribution (i.e. new model downloading) or online transfer learning (i.e. partial model updating) is needed. As illustrated in Figure C.1-1, an AI/ML model can be distributed from a NW endpoint to the devices when they need it to adapt to the changed AI/ML tasks and environments.

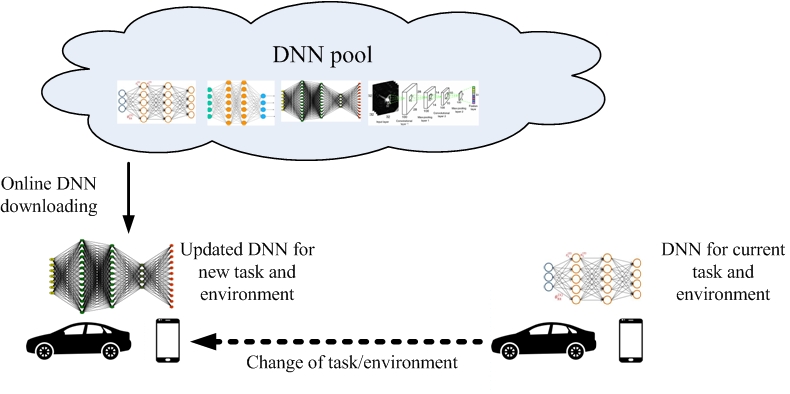


Figure C.1-1. AI/ML model downloading over 5G system

The model to be distributed can be determined in two ways: Requested by a device, or controlled by a network server. The condition of the first mechanism is that the device can make the model selection/re-selection decision based on the understanding to the oncoming AI/ML task, environment and the list of the models available at the network server. As shown in Figure C.1-2, a model selector on the device is trained to select the best DNN for different input data.

The model selector is trained to determine the optimum DNN model for a new, unseen input using a set of automatically tuned features of the DNN model input, and taking into consideration the precision constraint and the characteristics of the input.

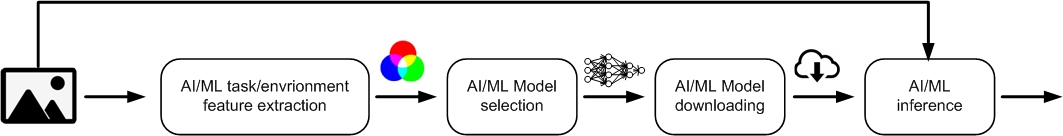


Figure C.1-2. AI/ML model selection and downloading

The data rate for downloading the needed models depends on the following factors:

* Size of the model

This depends on different AI/ML applications. Along with the increasing performance requirements to AI/ML operations, the sizes of the models also keep increasing, although model compression techniques are under improvements.

* Required downloading latency

This depends on how fast the model needs to be ready at the device. It is impacted by the extent to which the oncoming application can be predicted. Considering the unpredictability of user behaviour and typical waiting time a user can tolerate, the downloading of the AI/ML model needs to be finished in seconds or even in milliseconds. Different from a streamed video which can be played when a small portion is buffered, a DNN model can only be used until the whole model is completely downloaded.

In should be noted that the network-based and split AI/ML inference often requires a high and constant uplink data rate for continuously offloading sensing/intermediate data to the cloud/edge server. On the contrary, AI/ML model distribution mainly requires a high downlink data rate in a burst. This makes model distribution more suitable to the downlink-dominant (e.g. employing a high DL-to-UL ratio) mobile communications systems or systems with an unstable coverage. Of course the condition is that the mobile device’s computation resource can afford the on-board execution of the AI/ML model. If the computation load is beyond the device’s capability, the network-based or split inference has to be adopted.

Annex D:  
General principle of Distributed/Federated Learning over 5G system

With continuously improving capability of cameras and sensors on mobile devices, valuable training data, which are essential for AI/ML model training, are increasingly generated on the devices. For many AI/ML tasks, the fragmented data collected by mobile devices are essential for training a global model. In the traditional approaches, the training data gathered by mobile devices are centralized to the cloud datacenter for a centralized training.

However, an AI/ML model training often requires a large data set and significant computational resources for multiple weight-update iterations. Nowadays, most of the AI/ML model training tasks are performed in the power cloud datacenters since the resource consumption of the training phase significantly overweights the inference phase. In many cases, training a DNN model still takes several hours to multiple days. However, cloud-based training means that the enormous amount of training data should be shipped from devices to the cloud, incurring prohibitive communication overhead as well as the data privacy pressure at the network side [10]. Similar to the split AI/ML inference introduced in Annex B, AI/ML model training tasks can also work in a cloud-device coordination manner. Distributed Learning and Federated Learning are examples in this manner.

In Distributed Learning mode, as shown in Figure D.1-1, each computing node trains its own DNN model locally with local data, which preserves private information locally. To obtain the global DNN model by sharing local training improvement, nodes in the network will communicate with each other to exchange the local model updates. In this mode, the global DNN model can be trained without the intervention of the cloud datacenter [10].



Figure D.1-1. Distributed Learning

In Federated Learning (FL) mode, the cloud server trains a global model by aggregating local models partially-trained by each end devices. The most agreeable Federated Learning algorithm so far is based on the iterative model averaging [30]. As depicted in Figure D.1-2, within each training iteration, a UE performs the training based on the model downloaded from the AI server using the local training data. Then the UE reports the interim training results (e.g., gradients for the DNN) to the cloud server via 5G UL channels. The server aggregates the gradients from the UEs, and updates the global model. Next, the updated global model is distributed to the UEs via 5G DL channels. Then the UEs can perform the training for the next iteration.

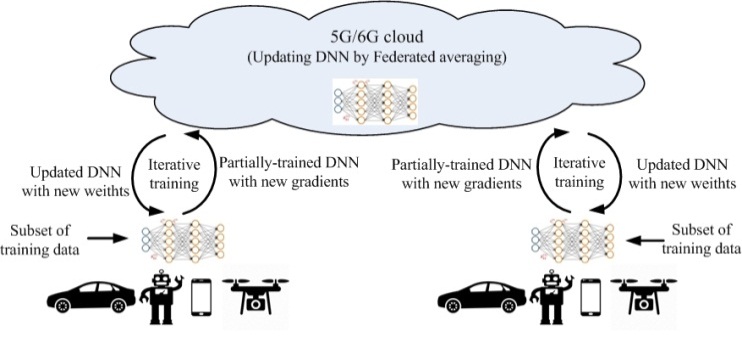


Figure D.1-2. Federated Learning over 5G system

The performance requirements for Distributed/Federated Learning are listed below. The requirements to 5G communication links (e.g. data rate, latency, reliability) can be derived from the following requirements.

* Training loss:

Training loss is the gap between the correct outputs and the outputs computed by the DNN model which indicates how well the trained DNN model fits the training data. Aim of training task is to minimize the training loss. Training loss is mainly affected by the quality of the training data and the efficiency of the training methods, i.e. whether the meaning of training data can be fully and properly explored. For Federated Learning, only when the valuable local training data can be fully learned in the duration of the iteration and the local training updates can be correctly reported to the cloud server within the target duration, the training loss can be minimized.

This implies that the requirements to the devices joining in the training process on the achievable UL data rate, latency and reliability for reporting the trained updates, and the achievable UL data rate, latency and reliability for distributing the model for training in next iteration. And to minimize the training loss with device heterogeneity (in computation and communication performance), training device selection and training configuration are needed before the training is performed in an iteration [31, 48] (will be introduced later in this section). The QoS of the relevant controlling messages, e.g. for training request, training resource reporting, training device selection, training configuration, and resource allocation for the training updates reporting, also needs to be guaranteed.

* Training latency:

Training latency is one of the most fundamental performance metrics of AI/ML model training task since it directly influences when the trained model is available for use. Nowadays, cloud-based training often takes several hours to multiple days. The latency of the Distributed/Federated Learning process would take even a longer time if the computation latency or the communication latency is not minimized.

The latency of the Distributed/Federated Learning process is determined by the convergence rate (e.g. number of iterations before the training process converges to a consensus) and the latency of each iteration which consists of computation latency and communication latency. The computation latency depends on the computation/memory resource available on training devices. The computation latency depends on the DL data rate available for model distribution and UL data rate available for trained model updating. The latency of the whole training process is determined by the larger one between the computation latency and the communication latency. Hence the latencies of the computation and communication links need to be cooperatively minimized. If the communication latency cannot match to the computation latency, the communication link will become the bottleneck and prolong the whole training process.

For synchronous Federated Learning, in each iteration, the training latency is determined by the last device that reports its training update because the federated aggregation can be finished when all needed training updates are correctly gathered. That means the device heterogeneity (in computation and communication performance) will also highly impact the overall training latency. Rather than requiring the UL transmission latency of a specific device, the overall latency required for all training devices to upload the training updates (device-group latency) needs to be defined. And the QoS of the controlling messages for minimizing the device-group latency, e.g. for training request, training resource reporting, training device selection, training configuration, and resource allocation for the training updates reporting, also needs to be guaranteed.

Energy efficiency:

For Distributed/Federated Learning, both the computation and communication processes consume considerable energy. The Federated Learning architecture and protocol should also consider the power constraints on the training devices and the energy efficiency on device as well as the network side.

* Privacy:

When training the DNN model by using the data originated at a massive of end devices, the raw data or intermediate data should be transferred out of the end devices. Compared to reporting it to the cloud/edge server, preserving privacy at the end devices can reduce the pressure of privacy protection at network side. For example, Federated Learning is an agreeable approach to avoid uploading the raw data from device to network, as a cloud-based training requires.

Annex E:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2020-09 | SA1#91-e | S1-203150 |  |  |  | TR Skeleton agreed | 0.0.0 |
| 2020-09 | SA1#91-e | S1-203382 |  |  |  | Agreements in SA1#91-e: S1-203137, S1-203284, S1-203074, S1-203135, S1-2031356, S1-203312, S1-203133, S1-203314, S1-203315, S1-203316, S1-203317, S1-203318, S1-203144, S1-203145, S1-203146, S1-203158 | 0.1.0 |
| 2020-11 | SA1#92-e | S1-204352 |  |  |  | Agreements in SA1#92-e: S1-204014, S1-204339, S1-204394, S1-204395, S1-204396, S1-204397, S1-204398, S1-204399, S1-204400, S1-204401, S1-204402, S1-204403, S1-204404 | 0.2.0 |
| 2021-03 | SA1#93-e | S1-210303 |  |  |  | Agreements in SA1#93-e: S1-210014, S1-210015, S1-210267, S1-210321, S1-210330, S1-210331,[S1-210332](file:///C:\Users\almodovarchicojl\Desktop\TSGS1_93e_Electronic_Meeting\Docs\S1-210332.zip), S1-210333, S1-210334, S1-210336, S1-210337, [S1-210374](file:///C:\Users\sultan\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\4XT0V618\docs\S1-210374.zip), [S1-210375](file:///C:\Users\sultan\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\4XT0V618\docs\S1-210375.zip), S1-210376, S1-210377, S1-210378, S1-210379, S1-210380, S1-210381, S1-210382, S1-210383, S1- 210384, S1-210385, S1-210386, S1-210387, S1-210388, S1-210389, S1-210390, S1-210391, S1-210409, S1-210410 | 0.3.0 |
| 2021-05 | SA1#94-e | S1-211303 |  |  |  | Agreements in SA1#94-e: S1-211075, S1-211254, S1-211396, S1-211397, S1-211398, S1-211399, S1-211400, S1-211401, S1-211402, S1-211403, S1-211404, S1-211405, S1-211406, S1-211407, S1-211408, S1-211409, S1-211410, S1-211411, S1-211412, S1-211413 | 1.1.0 |
| 2021-06 | SA#92e | SP-210507 |  |  |  | Raised by MCC to v.2.0.0 for Approval | 2.0.0 |
| 2021-06 | SA#92e | SP-210507 |  |  |  | Raised by MCC to v.18.0.0 following Approval | 18.0.0 |
| 2021-06 | SA#92e | SP-210507 |  |  |  | Corrected logo (5G->5GA) | 18.0.1 |
| 2021-09 | SA#93e | SP-211040 | 0001 | 1 | B | FS\_AMMT update to Functional requirements for AMMT services | 18.1.0 |
| 2021-09 | SA#93e | SP-211040 | 0002 | 1 | B | FS\_AMMT update to clause 7.4 Group performance Flocking Use Case | 18.1.0 |
| 2021-12 | SP-94 | SP-211496 | 0003 | 1 | D | FS\_AMMT update to CPR tables with new format | 18.2.0 |