 ISO/IEC JTC 1/SC 29/WG 04 N0427

**ISO/IEC JTC 1/SC 29/WG 04  
MPEG Video Coding   
Convenorship: CN**

**Document type:** Output Document

**Title:** Common Test and Training Conditions for FCM

**Status:** Approved

**Date of document:** 2023-10-20

**Source:** ISO/IEC JTC 1/SC 29/WG 04

**Expected action: None**

**Action due date: None**

**No. of pages:** 22 (without cover page) + XLS file

**Email of Convenor:** yul@zju.edu.cn

**Committee URL:** <https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-4>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 04 MPEG VIDEO CODING**

**ISO/IEC JTC 1/SC 29/WG 04 N** **0427**

**October 2023, Hannover, DE**

|  |  |
| --- | --- |
| **Title** | **Common Test and Training Conditions for FCM** |
| **Source** | **WG 04, MPEG Video Coding** |
| **Status** | **Approved** |
| **Serial Number** | **23375** |

**Table of Contents**

[1 Expected action: None 1](#_Toc150417432)

[2 Action due date: None 1](#_Toc150417433)

[3 Introduction 4](#_Toc150417434)

[3.1 Datasets and tasks 4](#_Toc150417435)

[3.2 Configurations 6](#_Toc150417436)

[3.3 Dataset details 7](#_Toc150417437)

[3.3.1 Tencent Video Dataset (TVD) 7](#_Toc150417438)

[3.3.2 OpenImages v6 7](#_Toc150417439)

[3.3.3 SFU-HW-Object-v1 7](#_Toc150417440)

[3.3.4 HiEve videos 9](#_Toc150417441)

[3.4 Network and split point details 9](#_Toc150417442)

[3.5 Cross-checking of results 10](#_Toc150417443)

[3.6 Additional required information from proposals 11](#_Toc150417444)

[3.6.1 Inference information 11](#_Toc150417445)

[3.6.2 Training information 11](#_Toc150417446)

[4 Training conditions 13](#_Toc150417447)

[4.1 Introduction and scope 13](#_Toc150417448)

[4.2 Common training datasets 13](#_Toc150417449)

[4.2.1 2.1. Datasets for detection and segmentation training 13](#_Toc150417450)

[4.2.2 2.2 Dataset for pedestrian tracking training 13](#_Toc150417451)

[4.3 Frameworks 14](#_Toc150417452)

[4.3.1 Detectron2 14](#_Toc150417453)

[4.3.2 JDE 14](#_Toc150417454)

[4.4 Weights check 15](#_Toc150417455)

[5 References 15](#_Toc150417456)

[Appendix A: Anchor metrics 17](#_Toc150417457)

[C.1 Bitrate measurement 17](#_Toc150417458)

[C.2 Task: Object Tracking 17](#_Toc150417459)

[C.3 Task: Instance segmentation / Object detection 17](#_Toc150417460)

[C.4 Runtime Measurement 18](#_Toc150417461)

[Appendix E Feature Anchor generation procedure 20](#_Toc150417462)

[E.1 Feature anchor generation 20](#_Toc150417463)

[E.3 Anchor generation for object tracking task: TVD videos and HiEve videos (feature) 23](#_Toc150417464)

[E.4 Anchor generation for SFU-HW videos for object detection (feature) 23](#_Toc150417465)

[Appendix F Anchor generation environment 24](#_Toc150417466)

[F.1 FCTM recommended environment for simulation 24](#_Toc150417467)

# Introduction

This document presents Common Test and Training Conditions (CTTC) for Feature Coding for Machines (FCM). This CTTC is based on the CfP [1] test conditions. In particular, proposals are required to provide results with task result meeting a set of performance points (PPs) that define an acceptable range of task result (mAP or MOTA) value, so that BD-rate computations are performed on overlapping curves. Compared to the CfP test conditions, only the top four PPs are retained. For PP1 and PP2, cross-check acceptance is reported based on the maximum and minimum for PP0 and PP1, respectively. However, where PP1 and PP2 fall outside their respective acceptance ranges the count of acceptable performance points is highlighted yellow.

## Datasets and tasks

Table 1 lists the datasets and tasks used in this CTC and defines the performance-point range of acceptable task result for each case.

Table . Acceptance range for task performance point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset – task** | **Minimum boundary** | **Maximum boundary** | **Granularity** |
| OpenImages – object detection | PP – 3 | PP + 2 | Dataset |
| OpenImages – instance segmentation | PP – 3 | PP + 2 | Dataset |
| SFU – object detection | PP – 3 | PP + 2 | Class |
| TVD – object tracking | PP – 3 | PP + 2 | Class |
| HiEve – object tracking | PP – 3 | PP + 2 | Class |

Figure **1** shows an example of the acceptance ranges for each performance point for TVD object tracking. Note that for the BD-rate computation to produce a result, the contribution results must be monotonic. For PP1 and PP2, results within the wider (yellow) range are accepted but the cross-check summary will show a yellow background should results in the yellow region be encountered.



Figure 1. Example of acceptance ranges for TVD object tracking performance points.

Video datasets are analyzed at the class level and FCTM includes scripts capable of synthesizing class-level results from sequence results.

The performance point (task result) targets are defined in Table 2, Table 3, and Table 4.

Table  Task results on OpenImages and TVD

|  |  |  |  |
| --- | --- | --- | --- |
| Task performance point (PP) | OpenImages | | TVD |
| Instance segmentation: mAP [%] | Object detection: mAP [%] | Object tracking: MOTA [%] |
| OVERALL |
| PP0 | 80.682 | 78.871 | 49.84 |
| PP1 | 79.187 | 78.056 | 47.11 |
| PP2 | 77.049 | 77.051 | 43.18 |
| PP3 | 72.618 | 74.755 | 35.76 |

Table 3 Object tracking results on HiEve

|  |  |  |
| --- | --- | --- |
| HiEve | | |
| Object tracking: MOTA [%] | | |
|  | HIEVE-1080P | HIEVE-720P |
| PP0 | 31.12 | 35.75 |
| PP1 | 30.39 | 34.90 |
| PP2 | 29.29 | 33.66 |
| PP3 | 27.42 | 31.70 |

Table 4 Object detection results on SFU-HW

|  |  |  |  |
| --- | --- | --- | --- |
| SFU-HW | | | |
| Object Detection: mAP [%] | | | |
|  | Class A/B | Class C | Class D |
| PP0 | 77.360 | 69.088 | 64.061 |
| PP1 | 75.304 | 66.334 | 62.900 |
| PP2 | 73.039 | 64.837 | 60.917 |
| PP3 | 66.894 | 55.014 | 54.627 |

## Configurations

The following config(s) are to be used for coding experiments:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Config** | **Details** |
| OpenImages | All Intra (AI) | - |
| SFU, TVD, HiEve | Low delay (LDP) | With intra period, same as would be used with a RA GOP32 and 1s access requirement. |

## Dataset details

### Tencent Video Dataset (TVD)

The Tencent Video Dataset (TVD) consists of 3 video sequences for object tracking. The three video sequences, TVD-01, TVD-02 and TVD-03 are used for the CTTC for video coding for machines. TVD-01, TVD-02 and TVD-03 have 3000, 636 and 2334 frames respectively, and the resolutions of the videos are 1920x1080. The files are in MP4 format. The dataset is provided and labeled by Tencent.

Detailed information can be found on <https://multimedia.tencent.com/resources/tvd>.

### OpenImages v6

OpenImages V6 is a large-scale dataset, consists of 9 million training images, 41,620 validation samples, and 125,456 test samples. Note that all images are already compressed.

In this CTTC, a subset of the OpenImages dataset is used for testing. A total of 5000 images were selected for object detection and another 5000 images were selected for instance segmentation. While there is an overlap between the two subsets, these are not identical.

More information on the dataset can be found on <https://storage.googleapis.com/openimages/web/index.html>.

For the machine vision task performance with the OpenImages dataset, mAP@0.5 shall be used.

The dataset is available using the following license text:

*The annotations are licensed by Google LLC under the* [*CC BY 4.0*](https://creativecommons.org/licenses/by/4.0/) *license. The images are listed as having a* [*CC BY 2.0*](https://creativecommons.org/licenses/by/2.0/) *license.* ***Note:*** *while we tried to identify images that are licensed under a Creative Commons Attribution license, we make no representations or warranties regarding the license status of each image and you should verify the license for each image yourself.*

### SFU-HW-Object-v1

SFU-HW-Object-v1 is a labeled video data with object labeled on raw video sequences. It has already been used for MPEG (HEVC). It can be used for compression and object detection simultaneously. The dataset is provided under the Creative Commons license BY 4.0 (CC BY 4.0). Videos and labels of this dataset can be found in the following links:

Video (source for FCVCM activity): [ftp://mpeg.org](ftp://mpeg.org/) (contact Feature Compression for Video Coding for Machines chairs for login information)

Video (original source): <ftp://hevc@mpeg.tnt.uni-hannover.de/testsequences/>

Label: <https://dx.doi.org/10.25314/7d8efc0a-3943-4738-b7a5-72badb04d765>

Clips (specific coded frame range) of each video are used, as follows:

**Class A:** Size 2560x1600p 30 fps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Original size, framerate** | **Duration** | **Used frames** | **Cropped area position** |
| Traffic | 2560x1600p 30 fps | 5s | 117 to 149 | Line 80,  Column 1200 |

**Class B:** Size 1920x1080p 24-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| Kimono | 24 | 10s | 207 to 239 |
| ParkScene | 24 | 10s | 207 to 239 |
| Cactus | 50 | 10s | 403 to 499 |
| BasketballDrive | 50 | 10s | 471 to 599 |
| BQTerrace | 60 | 10s | 403 to 499 |

**Class C:** Size 832x480p (WVGA) 30-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| BasketballDrill | 50 | 10s | 471 to 599 |
| BQMall | 60 | 10s | 403 to 499 |
| PartyScene | 50 | 10s | 403 to 499 |
| RaceHorses | 30 | 10s | 235 to 299 |

**Class D:** Size 416x240p (WQVGA) 30-60 fps

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **fps** | **Duration** | **Used frames** |
| BasketballPass | 50 | 10s | 471 to 599 |
| BQSquare | 60 | 10s | 403 to 499 |
| BlowingBubbles | 50 | 10s | 403 to 499 |
| RaceHorses | 30 | 10s | 235 to 299 |

Note that the videos “FourPeople”, “Johnny”, and “KristenAndSara”, known as ‘Class E’ are not used in this CTTC.

For the machine vision task performance with the SFU dataset, mAP@0.5 is used.

### HiEve videos

“Human In Events” (“HiEve”) is a labelled video dataset with labels for object tracking, post estimation, and action recognition. This CTTC uses the HiEve dataset for object tracking.

The HiEve video anchor package includes the video files in MP4 format, which is the input format for the HiEve feature anchor package. The five videos used from the HiEve dataset are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sequence number** | **Descriptive name** | **Resolution** | **Frame count** |
| 2 | Human in lab1 | 1280×720 | 4819 |
| 13 | hm\_in\_playground | 1920×1080 | 1416 |
| 16 | hm\_in\_road | 1920×1080 | 700 |
| 17 | hm\_in\_square2 | 1280×720 | 966 |
| 18 | hm\_in\_stair2 | 1280×720 | 1614 |

The feature anchor divides the five videos into two classes based on resolution as follows:

* HIEVE\_1080P: 13, 16.
* HIEVE\_720P: 2, 17, 18.

All frames of each video are to be coded and the frame rate is set as 30fps regardless of the indicated frame rate in the provided MP4/MOV file.

The HiEve feature anchor package includes scripts to compute classwise results for the HiEve informative video anchor using provided detections files in the HiEve video anchor package.

The HiEve dataset (videos and ground truth) are available from the FTP server.

**NOTE**: Use of the HiEve videos requires returning a signed copy of a license agreement to Shanghai Jiao Tong University (SJTU), a copy of which is included with the test material.

## Network and split point details

Datasets are tested with the networks and split points as described in Table 5, along with performance metrics.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dataset** | **Task** | **Network** | **Split point** | **Rate measure** | **Task measure** |
| OpenImages | Object detection | FasterRCNN-X101-FPN | P-layer (P2-P5) | BPP | mAP @ 0.5 |
| OpenImages | Instance segmentation | MaskRCNN-X101-FPN | P-layer (P2-P5) | BPP | mAP @ 0.5 |
| TVD | Object tracking | JDE-1088x608 | Darknet-53 | Kbps | MOTA |
| HiEve | Object tracking | JDE-1088x608 | Layers 75, 90, and 105 (“ALT1”) | Kbps | MOTA |
| SFU | Object detection | FasterRCNN-X101-FPN | P-layer (P2-P5) | Kbps | mAP @ 0.5 |

Table . Dataset/task/network/split-point combinations.

## Cross-checking of results

The cross-check for a proposal is considered successful if all of the following are met:

* Bitrates and task results produced by the cross-checker differ by no more than the amounts specified in Table 7.
* Each task result falls within a lower and upper bound relative to the respective task performance point (see section 8.7.1).

When a relative amount is specified for a given result (i.e., BPP, bitrate, mAP, or MOTA), the cross-check result rc and proponent result rp must satisfy:

When an absolute amount is specified, the cross-check result rc and proponent result rp must satisfy:

These checks are implemented in the accompanying result template and a summary of these checks is presented in the ‘CrosscheckSummary’ worksheet. Note that although the spreadsheet implements checks for BD-rate, these are not part of the cross-check result.

Table . Cross-check acceptance thresholds.

|  |  |
| --- | --- |
|  | **Acceptance criteria** |
| **BPP/bitrate comparison** | Within or equal to 0.1% (relative) |
| **Task comparison** | Within or equal to 1.5% (relative) OR 0.1 (absolute difference) |

## Additional required information from proposals

### Inference information

The information described below is required to be provided for the inference process for both encoding and decoding processes.

* **Network Visualization:** Graphical representation of the neural network
* **Param. Number**: Total numbers of parameters in the neural network.
* **Param. Precision**: Bits for storing one parameter. Additionally, use “I” for indicating an integer parameter and use “F” to indicate a floating-point number. For example, if the proposed method uses 16-bit integer to represent a parameter, you can report this information as “16 (I)”.
* **MAC (Kilo):** Number of multiply–accumulate (MAC) operations per pixel in the worst case for the inference stage, where the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. Since different size of input may influence the value, it is suggested to use 3840x2160 as the input frame size for unification.
* **Mem.T (MB):** Temporary memory. It denotes the memory used to store the output feature map for all intermediate layers (forward pass). Since different size of input may influence the value, it is suggested to use 3840x2160 as the input frame size for unification. For reporting Mem.T (MB) the calculation process is also suggested to be provided for crosschecking.
* **Patch Size**: The size of input to the neural networks during inference (patchW×patchH×patchT, e.g., 64x64x3) where applicable (e.g., when patch-wise processing of features is performed).

### Training information

When applicable, it is required to report and discuss the following information for the training process.

* **Epoch**: The number of complete passes through the training data (e.g., 100)
* **Batch Size**: The number of samples processed before the model is updated. (e.g., 4Kx16frames)
* **Training Time**: CPU and/or GPU (e.g., 48h) and hardware such as CPU/GPU model and count (if different to that used for inferencing).
* **Learning Curve:** Plot of the training loss and validation loss (or similar) versus the number of epochs
* **Training Sets**: If a pre-trained model is used, the source of the pre-trained model and its training sets should be reported in detail. The size (number of images or videos) used in each training dataset shall be reported.
* **Training Configuration per Rate-Distortion Point**: Any changes in the requested information used to generate different rate-distortion points

The following additional training information could help to better understand proposed neural network-based methods:

* **Number of Iterations:** number of gradient updates within an epoch
* **Patch Size**: size of input to the neural networks (patchW×patchH×patchT, e.g., 64x64x3) where applicable (e.g., when patch-wise training using features is performed).
* **Learning Rate**: The amount that the weights are updated during training (e.g., 5e-4)
* **Optimizer**: The algorithm used to change the attributes of proposed neural networks (e.g., ADAM)
* **Loss Function**: The function to calculate the model error during training and optimization (e.g., L1, L2, etc.)
* **Preprocessing**: (e.g., preprocessing procedure, normalization, cropping method, rotation, zoom etc.)

# Training conditions

## Introduction and scope

This document describes proposed Common Training Conditions for FCVCM, that covers:

* Defining a common set of datasets for training.
* Defining the frameworks used for training.
* Defining the parameters used during training.
* Defining the split points at which additional learned layers are added.
* Defining the procedure for crosschecking training.
* Performing a ‘weights check’ to ensure no unintended modifications to the NN part 1 and part 2 parameters occurred during training.

These common training conditions are defined for following machine learning tasks:

* Object detection
* Instance segmentation.
* Pedestrian tracking.

## Common training datasets

### 2.1. Datasets for detection and segmentation training

The **Openimages dataset** [2] contains annotations and images. The annotations are licensed by Google LLC under [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/) license. The images are listed as having a [CC BY 2.0](https://creativecommons.org/licenses/by/2.0/) license. There are seven versions of the Openimage dataset from version 1 to version 7, version 6 and version 7 are used in FCVCM. In FCVCM, there are 47 classes that need to be detected or segmented. The Openimage version 7 contains about 1.7 million images, 1442545 images with no usable annotations for segementation and detection, 300497 images with bounding boxes, the number of bounding boxes coresponding for these 47 classes is 597869.

The Openimagev7 dataset can be downloaded using a third party library named *fiftyone*. To convert the ground truth format of the Openimagev7 dataset from the openimage ground truth format to the COCO ground truth format, a script openimages\_objseg\_convert\_cfp.py is used. To used this script, it is needed to provide a list of classes as used in the COCO ground truth format.

### 2.2 Dataset for pedestrian tracking training

The **HiEve dataset** [5] contains 19 sequences, five of them (s2, s13, s16, s17, s18) are used for testing, so the remaining contains 14 sequences can be used for training. These 14 sequences are used for training a model at the ALT1 split point, that model is used for test the five HiEve sequences.

The **PedTrackPP dataset** [4] contains many short sequences downloaded from the Pixabay website and Pexels website, and ground truths for each video.

Ground truth for pedestrian tracking contains bounding boxes and their corresponding tracking ids. It is not required that every identity or bounding box be annotated with atracking id. For bounding boxes do not have corresponding tracking id, the value of their tracking id isset to “0“.

The PedTrackPP dataset is used for training a model at the DN53 split point, that model is used for test the TVD dataset.

## Frameworks

Example scripts for training the detection and segmentation tasks using the Detectron2 framework [1] are attached.

### Detectron2

In the Detectron2 framework, the COCO format [2] for ground truth is used. So, if using the Openimages dataset in the Detectron2 framework, the format for ground truth needs to be converted from the Openimage annotation format to the COCO annotation format. A script for performing this conversion is included.

To train layers inserted into the FasterRCNN network, a script det2\_msfc\_det\_cfp.py is used. A command line example as below:

det2\_msfc\_det\_cfp.py --dataset oiv7train --network msfc\_plain --fasterrcnn train

To train layers inserted into the MaskRCNN network, a script det2\_msfc\_cfp.py is use. The following command line is an example:

det2\_msfc\_cfp.py --dataset oiv7train --network msfc\_plain train

For each sccript, command-line options are as follows:

--dataset: name of the dataset registered in the training file. The dataset used in the segmentation and detection task here is the OpenImage dataset

--network: the MSFC network is used and configed in the msfc\_cfp\_common.py file.

fasterrcnn: indicate that the cfg.MODEL.MASK\_ON option is off in the training file. This option is used for detection task using FasterRCNN network and pretrained weights file.

train: indicate that the task is training. Other options are test for doing an inferencing and visualise for doing a visualiser.

### JDE

A script for pedestrian tracking training using the JDE framework [8] is attached.

Pretrained weights: jde.1088x608.uncertainty.pt

The following command line is an example shows how to train an MSFC network inserted into the JDE framework:

python3 train\_cfp.py --cfg cfg/yolov3\_1088x608.cfg --data-cfg data/pexels\_pixabay.json --batch-size 32 --epoch 30 --lr 0.001 --unfreeze-msfc --network msfc\_plain

Command-line options used with this script are as follows:

--cfg: directory to the config file. In this example, the config had used for training the jde.1088x608.uncertainty.pt weights file is used.

--data-cfg: training dataset config

--batch-size, --epoch, --lr: batch-size, number of epochs, and learning rate used during the training process

unfreeze-msfc: this option indicate that the MSFC added in the network will be trained

--network: the MSFC variant is used

## Weights check

Each checkpoint file contains all parameters for NN parts 1 and 2 from the pretrained weights file, in addition to parameters for the inserted MSFC layers.

The pretrained weights file includes convolutional layer weights, bias weights and may include some other types of parameter such as running\_mean, running\_var from batch normalization layers.

To compare two weights files, run the following command line:

compare\_weights.py --wf1 /path/to/weights1.pth --wf2 /path/to/weights2.pth

where --wf1 option is the directory to the first weights file and –wf2 option is the directory to the second weights file. The output of the above task is the differences of all weights in the first and the second weights file.

A list of common module names is printed along with the detected differences in the parameters of each common module. By comparing a modified network with the original network, the common module names will be all the NN part 1 and 2 layers.

When comparing with the pretrained weights file, the differences of thes common module names (i.e., NN parts 1 and 2) need to all be 0.0.

Any inserted module names are not present in the pretrained weights file and hence are omitted from this check.

NOTE: If weights are improperly frozen during training, weights that are extracted and supplied to FCTM are likely to result in severe performance degradation.

# References

1. WG2, “Updated Call for Proposals on Feature Compression for Video Coding for Machines”, N00319, October 2023.
2. WG2, “Requirements and Use cases for Video Coding for Machines”, N00190, April 2022.
3. Y. Wu, A. Kirillov, F. Massa, et al. "Detectron2,", https://github.com/facebookresearch/detectron2
4. Z. Wang, L. Zheng, Y. Liu, et al. "Towards real-time multi-object tracking," in European Conference on Computer Vision (ECCV). 2020: 107-122.

[1] https://github.com/facebookresearch/detectron2

[2] <https://storage.googleapis.com/openimages/web/index.html>

[3] <https://cocodataset.org/#home>

[4] R.Nguyen, C. Rosewarne (Canon), “[FCVCM] Suggested object tracking ground truth for selected “Pixabay” and “Pexels” videos”, m64558, July 2023

[5] Weiyao Lin, Kaiyuan Dong, Aixin Zhang, “[VCM] License for HiEve dataset””, m55774, Jan 2021

[8] https://github.com/Zhongdao/Towards-Realtime-MOT

Appendix A: Anchor metrics

## A.1 Bitrate measurement

For each image dataset, bits per pixel (BPP) shall be used. BPP is the number of bits occupied by each pixel, which is defined by:

refers to the total number of bits overall images and refers to the total number of pixels overall images at their original resolution.

For each video sequence, the bitrate shall be measured in kilobits per second (kbps). This is defined as:

Here refers to the total number of bits of the whole video sequence, *fps* denotes the number of frames per second of the video sequence and *frames* denote the number of encoded frames of the video sequence.

For classwise bitrate reporting, the sequence-length weighted average bitrate of the calculated sequence bitrates is to be reported. This is defined as:

Here *lengthCLS* refers to the total length of sequneces in the class in seconds, *bitrateCLS* refers to the classwise bitrate, *bitraten* refers to the bitrate of sequence *n* in class CLS containing *N* sequences, *FramesToBeEncodedn* refers to the count of frames to be encoded for sequence *n* and *FrameRaten* refers to the fame rate of sequence *n*.

## A.2 Task: Object Tracking

For the object tracking task, Multiple Object Tracking Accuracy (MOTA) [1] shall be used to measure performance.

The MOTA accounts for all object configuration errors made by the tracker, false positives, misses (true negative), mismatches, and overall frames.

where , , and are the number of false negatives, the number of false positives, the number of mismatch error (ID Switching between 2 successive frames), and the number of objects in the ground truth respectively at time .

## A.3 Task: Instance segmentation / Object detection

For both object detection and instance segmentation, mean Average Precision (mAP) [2] [3] shall be used to measure the performance of the network.

For a given category of object, true positive , false positive , false negative , and true negative are defined with an Intersection over Union (IoU) threshold for that category, where true/false represents the output of the neural network, positive/negative represents the label in the ground truth.

Then, recall of the given IoU threshold is defined as the proportion of all true positive examples in all true positive and false negative examples corresponding to that IoU threshold:

The precision of the given IoU threshold is the proportion of all true positive examples which are from all positive examples:

A neural network of segmentation may achieve several pairs of recall and precision values corresponding to a certain IoU threshold and different confidence levels. For each recall value in the pairs, let takes the maximum precision value in all precision values for which the corresponding recall values are above the given recall value :

Average Precision (AP) of a given category of object is defined as the average value of for all recall values provided by the neural network, which can characterize the area of the entire precision-recall curve.

Mean Average Precision (mAP) is an averaged AP over all categories of objects and in a range of IoU thresholds.

The following mAP variant is used:

* [mAP@0.5](mailto:mAP@0.5): the mAP when the IoU threshold is 0.5.

## A.4 Runtime Measurement

Runtime includes Encoding time (EncT), Decoding time (DecT) and Task time for part 1 and part 2 of the network (TaskT1, TaskT2) for complexity measurement. The proposed runtime measurements for a FCVCM solution are:

* **TaskT1**: Time needed to perform part 1 of the network (e.g., the backbone) to produce features.
* **EncT:** Time needed to convert feature input to bitstream.
* **DecT:** Time needed to convert bitstream to decoded features.
* **TaskT2**: Time needed to perform part 2 of the network (e.g., the head) to complete the task based on the decoded features.

For the purpose of reporting encoding and decoding running times, the feature anchor and proposal should be simulated on the same platform, e.g., the same CPU, to have reliable runtime comparison.

For summary runtime reporting, an “overall” result averaging the classwise runtimes from each dataset excluding the HiEve dataset is computed.

For video datasets, the NN part 1 and NN part 2 runtimes are included in the reported ‘EncT’ and ‘DecT’ (and thus contribute to the reported encoder and decoder runtime ratios) in the attached revised result template.

This is a workaround for the current video feature anchors, which do not separate NN part 1/2 runtimes from the feature anchor runtimes, and will be addressed in future revisions of the video feature anchors (after the CfP).

Appendix B Feature Anchor generation procedure

## B.1 Feature anchor generation

FCTM-v1.0.0 includes scripts to launch CompressAI-Vision, with FCTM instantiated as the feature codec.

The scripts are located within the FCTM repository, as follows:

* scripts/evaluation/hieve/fctm\_eval\_on\_hieve\_tracking.sh
* scripts/evaluation/mpeg\_oiv6/fctm\_eval\_on\_mpeg\_oiv6.sh
* scripts/evaluation/sfu\_hw\_obj/fctm\_eval\_on\_sfu\_hw\_obj.sh
* scripts/evaluation/tvd/fctm\_eval\_on\_tvd\_tracking.sh

Each script is run once per sequence and per QP, and performs the following operations:

1. NN part 1
2. Feature encoding
   1. Feature reduction
   2. Feature conversion
   3. Inner coding (encoding)
3. Feature decoding
   1. Inner coding (decoding)
   2. Feature inverse conversion
   3. Feature restoration
4. NN part 2
5. Evaluation (sequence level)

By default, low-delay inner encoding is performed as a set of parallel encoding processes, divided based on the intra-period used in the FCTM CTTC. NN part 1 and 2 jobs exploit parallelism built in the Pytorch (when running on CPU). Accordingly, parallelism for video datasets is directly supported by CompressAI-Vision.

For image datasets, coded using all-intra, the dataset may be encoded as a set of parallel jobs by dividing the dataset into chunks via:

* ++pipeline.codec.skip\_n\_frames=<start\_frame>
* ++pipeline.codec.m\_frames\_to\_be\_encoded=<frame\_count>

Note that the intra period for VTM is configured as if a GOP32 random-access picture structure were being used. In particular, intra period settings for CTTC sequence frame rates are enumerated below:

* + For frame rate equal to 20fps, 24fps, 25fps, or 30fps, use value 32.
  + For frame rate equal to 50fps or 60fps, use value 64.
  + For frame rate equal to 100fps, use value 96.

Table 2 shows the configurations that need to be indicated for each test dataset, and QPs applied for FCTM-v1.0.0.

NOTE: Revisions of the FCTM after v1.0.0 may alter the QPs, e.g., due to retraining, so that the feature anchor results remain within the target PP ranges specified in section 3.1.

Table 2. The configurations of the anchor for each dataset (for FCTM-v1.0.0)

|  |  |  |  |
| --- | --- | --- | --- |
| **Machine Task** | **Configuration** | **QP** | |
| Object detection on OpenImages | AI | 29, 31, 32, 39 | |
| Instance segmentation on OpenImages | AI | 27, 32, 37, 39 | |
| Object tracking on TVD | LDP | TVD-01 | 25, 30, 33, 38 |
| TVD-02 | 25, 30, 35, 37 |
| TVD-03 | 28, 33, 34, 36 |
| Object Detection on SFU | LDP | Traffic | 24, 28, 29, 34 |
| Kimono | 27, 35, 37, 39 |
| ParkScene | 18, 22, 32, 36 |
| Cactus | 41, 47, 49, 51 |
| BasketballDrive | 22, 27, 32, 37 |
| BQTerrace | 22, 27, 29, 32 |
| BasketballDrill | 22, 27, 32, 39 |
| BQMall | 27 ,32, 37, 39 |
| PartyScene | 20, 30, 32, 39 |
| RaceHorsesC | 26, 32, 37, 41 |
| BasketballPass | 27, 32, 35, 39 |
| BQSquare | 22, 27, 30, 34 |
| BlowingBubbles | 27, 35, 36 ,37 |
| RaceHorses | 22, 27, 37, 41 |
| Object tracking on HiEve | LDP | 2 | 24, 27, 30, 33 |
| 13 | 17, 20, 24, 25 |
| 16 | 21, 23, 25, 27 |
| 17 | 17, 19, 21, 26 |
| 18 | 20, 23, 25, 30 |

Appendix C Anchor generation environment

## C.1 FCTM recommended environment for simulation

Software package versions used to generate the feature anchors are as follows:

* Python 3.8
* CUDA 11.8
* PyTorch 2.0.0
* Torchvision 0.15.1
* ffmpeg 4.2.2 (for video conversion)
* VTM-12.0 software (for feature anchor inner coding)